Improving Bayesian Neural Networks by Adversarial Sampling

Jiaru Zhang[1], **Yang Hua**[2], **Tao Song**[1], **Hao Wang**[3], **Zhengui Xue**[1], **Ruhui Ma**[1], **Haibing Guan**[1]
[1]Shanghai Jiao Tong University [2]Queen's University Belfast [3]Louisiana State University
{jiaruzhang,songt333, zhenguixue, ruhuima, hbguan}@sjtu.edu.cn,
Y.Hua@qub.ac.uk, haowang@lsu.edu

Abstract

Bayesian neural networks (BNNs) have drawn extensive interest due to the unique probabilistic representation framework. However, Bayesian neural networks have limited publicized deployments because of the relatively poor model performance in real-world applications. In this paper, we argue that the randomness of sampling in Bayesian neural networks causes errors in the updating of model parameters during training and poor performance of some sampled models in testing. To solve this, we propose to train Bayesian neural networks with Adversarial Distribution as a theoretical solution. To avoid the difficulty of calculating Adversarial Distribution analytically, we further present the Adversarial Sampling method as an approximation in practice. We conduct extensive experiments with multiple network structures on different datasets, e.g., CIFAR-10 and CIFAR-100. Experimental results validate the correctness of the theoretical analysis and the effectiveness of the Adversarial Sampling on improving model performance. Additionally, models trained with Adversarial Sampling still keep their ability to model uncertainties and perform better when predictions are retained according to the uncertainties, which further verifies the generality of the Adversarial Sampling approach.

1 Introduction

Bayesian neural networks (Neal 2012; Gal 2016) provide a probabilistic view of deep learning frameworks. Variational Inference (VI) (Blundell et al. 2015; Blei, Kucukelbir, and McAuliffe 2017) is an effective method to train the Bayesian neural networks, especially in large-scale practical tasks. One of the primary advantages of Bayesian neural networks is that they can model both aleatoric and epistemic uncertainty due to the unique probabilistic representation of the network parameters. It has shown considerable potential and has been widely used in a variety of tasks, e.g., computer vision (Kendall and Gal 2017; Phan 2019; Gustafsson, Danelljan, and Schön 2020), natural language processing (Xiao and Wang 2019), active learning (Hernández-Lobato and Adams 2015), and reinforcement learning (Depeweg et al. 2017).

Bayesian neural networks have indeed few publicized deployments in industrial practice despite the theoretical advancements (Wenzel et al. 2020). Researchers have proposed multiple explanations and solutions for this phenomenon. Wenzel et al. revealed that the temperature of parameters and the prior selection have a clear influence on model generalization performance (Wenzel et al. 2020). However, they did not provide a clear method to improve the model performance. Initializing Bayesian neural networks with the parameters from pretrained deterministic neural networks has also been proposed to improve the performance (Krishnan, Subedar, and Tickoo 2020; Deng et al. 2020). Although their methods are effective in practice, it is still unknown that why Bayesian neural networks cannot learn a suitable representation and perform well on their own.

To address it in this paper, we first present an explanation for the phenomenon by examining the training and inference processes of Bayesian neural networks. We discover that there are some errors in the updating of model parameters during training and some sampled models with poor performance in testing because of the randomness of sampling. Further, we propose to train Bayesian neural networks with Adversarial Distribution based on the discovery, where the parameters are updated according to the likely sampled model with the worst performance. As the accurate calculation of the Adversarial Distribution is usually difficult, we propose the Adversarial Sampling method as an approximation and it is simple to implement in practice. We carry out experiments to validate our motivation and the effectiveness of the proposed Adversarial Sampling method on different datasets.

In summary, the main contributions are listed as follows:

- We argue that the randomness of sampling in Bayesian neural networks causes errors in updating parameters during training and models with poor performance during testing.
- We propose to train Bayesian neural networks with Adversarial Distribution. According to our theoretical analysis, it can improve the worst performance of the model in multiple samplings and enhance its predictive performance.
- To use the adversarial distribution training in practice, we further propose the Adversarial Sampling method as an approximation. It can be implemented easily in current Bayesian neural network frameworks.

- Experiments on multiple network structures and datasets verify our theoretical analysis and the effectiveness of the Adversarial Sampling method. We release our codes at `https://github.com/AISIGSJTU/AS`.

2 Related Work

2.1 Bayesian neural networks

Unlike deterministic deep neural networks where a point estimate of model parameters is obtained by optimizing a specific objective function, Bayesian neural networks (Buntine 1991; Neal 2012; Blundell et al. 2015) are trained to find the posterior distribution of the parameters instead of a point estimate. Variational Inference (Graves 2011; Foti et al. 2014; Blundell et al. 2015; Blei, Kucukelbir, and McAuliffe 2017) and Markov Chain Monte Carlo (Chen, Fox, and Guestrin 2014; Bardenet, Doucet, and Holmes 2017) are two mainstream methods to train Bayesian neural networks. Although Variational Inference scales better than the Markov Chain Monte Carlo approach and is gaining in popularity, most of the extant works fail to meet the aim of practicability and there are few publicized deployments in industrial practice.

Researchers have offered several explanations and remedies to the problem. Wenzel et al. demonstrate that cooling the posterior with a temperature T improves generalization performance considerably (Wenzel et al. 2020). However, the posterior probability becomes more concentrated on the set of point estimates with a cooler temperature T, i.e., degenerating into a deterministic neural network. From another point of view, Krishnan et al. suggest using the parameters from a pretrained deterministic neural network to establish the prior and initialize the model parameters (Krishnan, Subedar, and Tickoo 2020). Similarly, Deng et al. propose BayesAdapter as a tool for converting pretrained deterministic neural networks into Bayesian neural networks (Deng et al. 2020). Both methods are successful in increasing model performance, but they do not explain why the initialization from deterministic neural networks is essential and there is still a large gap for further improvement.

2.2 Adversarial Perturbation

Adversarial perturbation on data, i.e., Adversarial Examples, has been studied in detail (Goodfellow, Shlens, and Szegedy 2015; Sun, Tan, and Zhou 2018). A small but intentionally Adversarial Perturbation on input data can result in a significant shift in model output. Researchers have proposed many kinds of methods of Adversarial Perturbation on data, e.g., FGSM (Goodfellow, Shlens, and Szegedy 2015), BIM (Kurakin, Goodfellow, and Bengio 2017), CW (Carlini and Wagner 2017), PGD (Madry et al. 2018), and AutoAttack (Croce and Hein 2020). Moreover, using Adversarial Examples as training data, i.e., Adversarial Training (Goodfellow, Shlens, and Szegedy 2015), has also been used as an effective method to enhance the adversarial robustness of models. A worth mentioning method is Adversarial Distribution Training (Dong et al. 2020), which also formulates the adversarial perturbation as a probabilistic distribution. Our method differs from the Adversarial Distribution Training because we target improving the generalization performance of Bayesian neural networks, while theirs is used on improving the adversarial robustness of deterministic neural networks.

There are also several studies about the adversarial perturbation on model parameters. It has been shown that adversarial perturbation on weights improves the adversarial robustness of models by combining with the adversarial training (Wu, Xia, and Wang 2020). Furthermore, Zheng et al. demonstrate that the adversarial perturbation on model parameters alone serves as a regularization approach by enforcing the model finding flat local minima of the empirical risk, and enhances the model generalization (Zheng, Zhang, and Mao 2021). To some extent, our Adversarial Sampling approach can be viewed as a natural extension of these existing approaches on Bayesian neural networks.

3 Background

3.1 Bayesian Neural Networks with Variational Inference

Suppose we have observations $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \dots\}$, a Bayesian neural network parameterized by $\mathbf{W}$ uses a variational distribution $Q_\theta(\mathbf{W})$ to approximate the real posterior probability $P(\mathbf{W}|\mathcal{D})$. The Bayesian neural network is trained by minimizing the Kullback–Leibler (KL) divergence

$$\begin{aligned} KL(Q_\theta(\mathbf{W})||P(\mathbf{W}|\mathcal{D}))) &= -\int Q_\theta(\mathbf{W}) \log \frac{P(\mathbf{W}|\mathcal{D})}{Q_\theta(\mathbf{W})} d\mathbf{W} \\ &= \log P(\mathcal{D}) - \int Q_\theta(\mathbf{W}) \log \frac{P(\mathbf{W},\mathcal{D})}{Q_\theta(\mathbf{W})} d\mathbf{W}. \end{aligned} \tag{1}$$

Since $\log P(\mathcal{D})$ is a constant for given observations $\mathcal{D}$, minimizing the KL divergence is equivalent to minimizing

$$\begin{aligned} \mathcal{L} &= -\int Q_\theta(\mathbf{W}) \log \frac{P(\mathbf{W},\mathcal{D})}{Q_\theta(\mathbf{W})} d\mathbf{W} \\ &= \underbrace{-\mathbb{E}_{\mathbf{W}\sim Q_\theta(\mathbf{W})} \log P(\mathcal{D}|\mathbf{W})}_{\mathcal{L}_p} + \underbrace{KL(P(\mathbf{W})||Q_\theta(\mathbf{W}))}_{\mathcal{L}_r}. \end{aligned} \tag{2}$$

Note that $-\mathcal{L}$ is a lower bound of $logP(\mathcal{D})$, thus $\mathcal{L}$ is usually called the Evidence Lower Bound (ELBO) loss (Blei, Kucukelbir, and McAuliffe 2017). It can be divided into two terms. The first term is directly related to the predictions and it is named as the prediction loss $\mathcal{L}_p$. The second term can be seen as a regularization on the model parameters, and it is named as the regularization loss $\mathcal{L}_r$.

The target of the training process is to find the parameters θ of the variational distribution $Q_\theta(\mathbf{W})$ to minimize $\mathcal{L}$:

$$\theta = \underset{\theta}{\text{argmin}} \mathcal{L}. \tag{3}$$

The prediction $\mathbf{y}$ of given input $\mathbf{x}$ is obtained by multiple stochastic forward passes through sampling K times from the probability distribution:

$$p(\mathbf{y}|\mathbf{x},\mathcal{D}) \approx \frac{1}{K}\sum_{k=1}^{K} p(\mathbf{y}|\mathbf{x},\mathbf{W}_k), \mathbf{W}_k \sim Q_\theta(\mathbf{W}). \tag{4}$$

It is also called the Bayes ensemble, as it can be seen as an ensemble of multiple likely models (Deng et al. 2020).

3.2 Uncertainty Estimation

One of the important properties of Bayesian neural networks is the ability to estimate uncertainty, which has been used in various domains to enhance model performance, such as scene segmentation (Kendall and Gal 2017) and sentiment analysis (Xiao and Wang 2019). On the other hand, uncertainty estimation can also be used to detect anomalous samples or out-of-distribution samples (Smith and Gal 2018; Antorán 2019). For a classification model with parameters $\mathbf{W}$, input $\mathbf{x}$ and output y in classes $C = \{c_1, c_2, \ldots, c_m\}$, following previous approaches as follows (Antorán 2019; Depeweg 2019; Zhang et al. 2021), the total uncertainty of the prediction can be modeled by its predictive entropy

$$H(y|\mathbf{x}, \mathbf{W}) = \sum_{i=1}^{m} p(y = c_i|\mathbf{x}, \mathbf{W}) \log p(y = c_i|\mathbf{x}, \mathbf{W}). \tag{5}$$

It contains both aleatoric uncertainty H_a and epistemic uncertainty H_e. The aleatoric uncertainty H_a is

$$H_a(y|\mathbf{x}, \mathbf{W}) = \mathbb{E}_{\mathbf{W}} H(y|\mathbf{x}, \mathbf{W}) \approx \frac{1}{K} \sum_{k=1}^{K} H(y|\mathbf{x}, \mathbf{W}_k). \tag{6}$$

Hence it can be estimated by K Monte Carlo samplings. The epistemic uncertainty H_e is given by the difference between the overall uncertainty H and the aleatoric uncertainty H_a, i.e.,

$$H_e(y|\mathbf{x}, \mathbf{W}) = H(y|\mathbf{x}, \mathbf{W}) - \mathbb{E}_{\mathbf{W}} H(y|\mathbf{x}, \mathbf{W}). \tag{7}$$

4 Adversarial Sampling

4.1 Explanation of the Poor Performance

As demonstrated in Equations (3) and (4), Monte Carlo sampling is usually used to sample from the Bayesian posterior throughout the training process, and the original parameters are subsequently updated based on the gradients and losses from the sampled models. Because the models utilized in training are randomly sampled from the Bayesian neural networks, there are some errors in the updating of model parameters. Figure 1 presents the error rates of Bayesian neural networks and deterministic neural networks with different random seeds during training. The curves of Bayesian neural networks fluctuate more sharply, which verifies that there are some errors in updating the parameters of Bayesian neural networks because of the randomness.

Similarly, the final predictions are derived from an ensemble of the forward passing of multiple random likely models during testing. Therefore, some models with poor performance are yielded from the random sampling process. Figure 2 shows the distribution of accuracies of sampled models from a Bayesian neural network. Some models have much lower accuracies compared with others, which confirms our analysis that there exist some models with poor performance in the random sampling process.

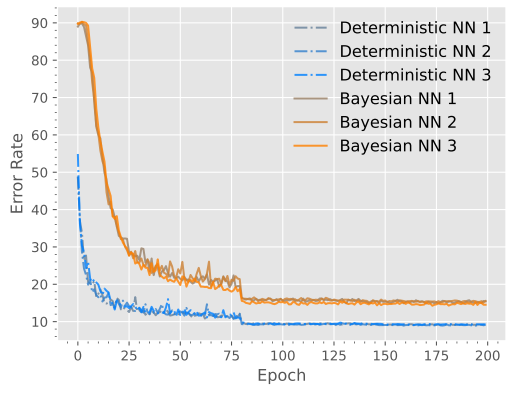


Figure 1: Error rates of Bayesian neural networks and deterministic neural networks with different random seeds during a 200-epoch training. The ResNet20 structure and dataset CIFAR-10 are used. *Best viewed in color.*

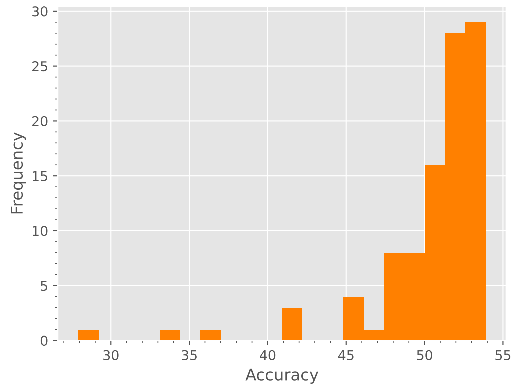


Figure 2: Accuracies of 100 sampled models from a Bayesian neural network. The ResNet56 structure and dataset CIFAR-100 are used. *Best viewed in color.*

4.2 Training with Adversarial Distribution

Based on our explanation of the poor performance, we propose to train Bayesian neural networks with Adversarial Distribution $Q_{adv}(\mathbf{W})$ instead of using original distribution $Q_\theta(\mathbf{W})$. Concretely, given a dataset $\mathcal{D}$ and the parameter distributions of a Bayesian neural network $Q_\theta(\mathbf{W})$, the corresponding Adversarial Distribution $Q_{adv}(\mathbf{W})$ is defined as follows:

$$Q_{adv} = \underset{W[Q_{adv}, Q_\theta] \le d}{\operatorname{argmax}} - \mathbb{E}_{\mathbf{W} \sim Q_{adv}(\mathbf{W})} \log P(\mathcal{D}|\mathbf{W}), \tag{8}$$

where d is a hyperparameter to control $W[Q_{adv}, Q_\theta]$. $W[Q_{adv}, Q_\theta]$ denotes the Wasserstein distance between Q_{adv} and Q_θ, which is a widely used metric between probabilistic distributions in deep learning (Frogner et al. 2015; Arjovsky, Chintala, and Bottou 2017; Cheng et al. 2020). It

Algorithm 1: Training with Adversarial Sampling

Input: Variational posterior parameters (μ, σ), Batch data $\mathcal{D}$
Parameters: Iterations N, Step length for perturbation α
Output: Updated variational posterior parameters (μ, σ)

```
 1: Sample $\epsilon_{adv} \sim \mathcal{N}(0, 1)$
 2: for sufficient iterations $N$ do
 3:     Let $w_{adv} = \mu + \epsilon_{adv} \cdot \sigma$
 4:     Calculate the adversarial loss $\mathcal{L}_{adv}$ with parameter $w_{adv}$ and data $\mathcal{D}$
 5:     Update $\epsilon_{adv} = \epsilon_{adv} + \alpha \cdot sign(\frac{\partial \mathcal{L}_p}{\partial \epsilon_{adv}})$
 6: end for
 7: Let $w_{adv} = \mu + \epsilon_{adv} \cdot \sigma$
 8: Calculate the adversarial loss $\mathcal{L}_{adv}$ with parameter $w_{adv}$ and data $\mathcal{D}$
 9: Sample $\epsilon \sim \mathcal{N}(0, I)$
10: Let $w = \mu + \epsilon \cdot \sigma$
11: Calculate the prediction loss $\mathcal{L}_p$ with parameter $w$ and data $\mathcal{D}$
12: Calculate the regularization loss $\mathcal{L}_r$ analytically
13: Calculate the total loss $\mathcal{L}$ with Equation (11)
14: Update parameter $\mu$ and $\sigma$ with the total loss $\mathcal{L}$
```

is defined as

$$W\left[Q_{adv}, Q_\theta\right] = \inf_{\gamma \in \Pi(Q_{adv}, Q_\theta)} \mathbb{E}_{(x,y)\sim\gamma}[d(x, y)], \quad (9)$$

where $\Pi\left(Q_{adv}, Q_\theta\right)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginal distributions are Q_{adv} and Q_θ, and $d(x, y)$ represents the distance between the two points x and y. In this paper, we use the l_∞ metric as the distance metric $d(x, y)$.

Corresponding to the Adversarial Distribution, the adversarial loss $\mathcal{L}_{adv}$ is defined as

$$\mathcal{L}_{adv} = -\mathbb{E}_{\mathbf{W}\sim Q_{adv}(\mathbf{W})} \log P(\mathcal{D}|\mathbf{W}). \quad (10)$$

The total learning target is formulated as

$$\theta = \underset{\theta}{\operatorname{argmin}} \left((1 - \lambda) \cdot \mathcal{L}_p + \lambda \cdot \mathcal{L}_{adv} + \mathcal{L}_r\right), \quad (11)$$

where λ is a hyperparameter to control the ratio of training with Adversarial Distribution. It will degenerate to the original target in Equation (3) when $d = 0$ or $\lambda = 0$. Intuitively, sampling from the Adversarial Distribution yields likely models with the worst performance in the Bayesian neural network. Therefore, we update the parameters according to the worst model to guarantee the performance of the regularly sampled models during prediction. In this way, it can also effectively reduce the performance disparity between the likely models.

4.3 Adversarial Sampling as an Approximation

Calculating the Adversarial Distribution Q_{adv} in Equation (8) analytically for Bayesian neural networks is usually difficult. Therefore, we propose the Adversarial Sampling technique as a practical approximation. To be more specific, we sample each parameter from the original parameter distribution Q_θ, which is a Gaussian distribution with a mean of μ and a variance of σ^2:

$$w_{adv} \sim \mathcal{N}(\mu, \sigma^2). \quad (12)$$

The parameter w is then adversarially perturbed using an iterative approach. The step below is repeated multiple times:

$$w_{adv} = w + \alpha \cdot \sigma \cdot sign(grad(w_{adv})), \quad (13)$$

where $grad(w_{adv})$ represents the gradient of parameter w of the corresponding loss function. The hyperparameter α is the step size of the adversarial perturbation. We adjust the scope of the adversarial perturbation using the standard deviation of the parameter σ, since a parameter with a larger standard deviation has higher randomness in regular sampling.

Denoting the iteration times as N, the total distance between w and w_{adv} satisfies $\|w - w_{adv}\| \leq N \cdot \alpha$. Therefore, it satisfies $W[Q_{adv}, Q_\theta] \leq d$ by setting $d = N \cdot \alpha$. Besides, as w_{adv} is generated by multiple iterations of increasing the prediction loss, it can be seen as a reasonable approximation of the argmax operation. Therefore, the process of generating the parameter w_{adv} can be regarded as a sampling from a distribution, and many w_{adv}s create an approximation of the Adversarial Distribution.

In practice, the parameter w is yielded by a random unit Gaussian noise $\epsilon \sim \mathcal{N}(0, 1)$: $w = \mu + \epsilon \cdot \sigma$ with the popularly used reparameterization trick (Blundell et al. 2015; Kingma, Salimans, and Welling 2015; Kingma and Welling 2014). Therefore, we just need to update the random noise ϵ with the same step size α instead of considering the standard deviation σ, making Adversarial Sampling simple to implement. The procedure to train Bayesian neural networks with Adversarial Sampling is provided in Algorithm 1.

5 Experiments

In this section, we empirically verify our motivation and investigate the effectiveness of the proposed Adversarial Sampling method. We train a variety of Bayesian neural networks, including ResNet20, ResNet56 (He et al. 2016), and VGG (Simonyan and Zisserman 2015), on CIFAR-10, and CIFAR-100 datasets (Krizhevsky 2009). For simplicity, we set the hyperparameter $\alpha = 0.02$ and $N = 5$ on models trained with Adversarial Sampling.

5.1 Verification of Motivation

We present the experimental results to verify that our analysis on the problem of Bayesian neural networks and our Adversarial Sampling method is effective to solve it. We train networks with and without Adversarial Sampling of structure ResNet20 on CIFAR-10 dataset with multiple random seeds and plot the error rates during the 200-epoch training in Figure 3. The error rates of models trained with Adversarial Sampling are obviously lower than that of original models, proving the effectiveness of the Adversarial Sampling method. The change trends of models trained with Adversarial Sampling are more stable and steady, which verifies

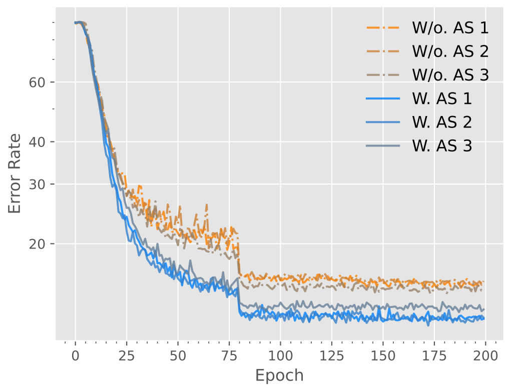


Figure 3: Error rates of models trained without Adversarial Sampling (AS) and with Adversarial Sampling with different random seeds during a 200-epoch training. *Best viewed in color.*

our motivation that training with Adversarial Sampling reduces the errors in the updating of model parameters during training.

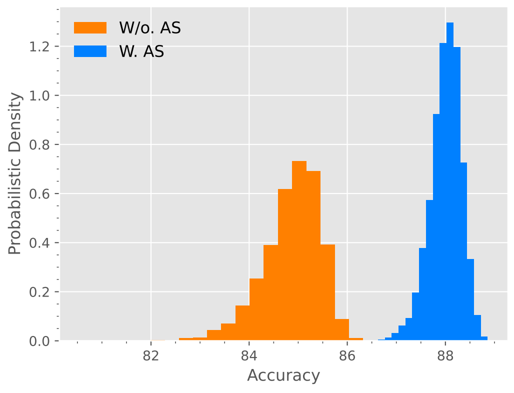
We sample 10000 likely models from the models trained with and without Adversarial Sampling and we record the test accuracy of them. The Bayesian neural networks of structure ResNet20 on CIFAR-10 dataset and Bayesian neural networks of structure ResNet56 on CIFAR-100 dataset are used. The results are shown in Figure 4. The models trained without Adversarial Sampling distribute more dispersed than the models trained with Adversarial Sampling, which verifies that the Adversarial Sampling method reduces the randomness of Bayesian neural networks. Additionally, the accuracy of the worst-performing model trained with Adversarial Sampling is clearly higher than that of the model trained without Adversarial Sampling.

5.2 Ablation Study

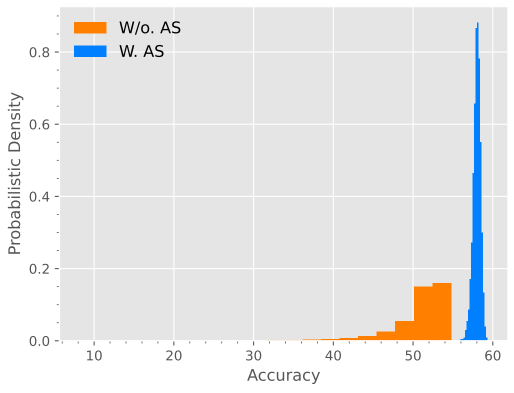
As indicated in Equation (11), the hyperparameter λ regulates the ratio of Adversarial Sampling during training. To evaluate the impact of different settings of the λ in practice, we measure the error rates of models trained with different λ throughout a 200-epoch training. The results are presented in Figure 5. The model performance is clearly improved when λ is gradually increased from 0 (i.e., a model trained without Adversarial Sampling) to 0.8. However, there is a significant drop in performance when λ reaches 1.0 (i.e., training with completely Adversarial Sampling). Therefore, the completely Adversarial Sampling harms the model performance and the introduction of parameter λ in Equation (11) is necessary.

5.3 Improvement on Model Performance

We present the classification accuracies in Table 1 to validate that the Adversarial Sampling method indeed enhances the model performance. The Bayesian neural networks of structure ResNet20, ResNet56, and VGG trained on CIFAR-10



(a) CIFAR-10, ResNet20



(b) CIFAR-100, ResNet56

Figure 4: Comparison between accuracies of 10000 sampled models from Bayesian neural networks trained without Adversarial Sampling (AS) and with Adversarial Sampling. *Best viewed in color.*

and CIFAR-100 datasets are used. We present three kinds of accuracies for each model: The lowest accuracy and the highest accuracy among 100 sampled models, and the accuracy of the ensembled model. The models trained with Adversarial Sampling have much higher lowest accuracies compared with the original models under all the conditions. It verifies our analysis that particular models with poor performance exist in normal training and the Adversarial Sampling method solves the problem. They also have clear higher highest accuracies and higher ensembled accuracies on all the network structures and datasets, which proves that the Adversarial Sampling method indeed improves the model performance.

5.4 Combination with Bayesian Fine-tune

As we mentioned above, initialization of Bayesian neural networks with the parameters from a trained deterministic neural network, i.e., Bayesian fine-tune, is an effective method to improve the performance of Bayesian neural net-

Dataset	Model	Lowest Accuracy	Highest Accuracy	Ensembled Accuracy
CIFAR-10	ResNet20	82.73 ± 0.88	86.03 ± 0.43	87.01 ± 0.65
	ResNet20 + AS	$\mathbf{86.33 \pm 0.45}$	$\mathbf{88.35 \pm 0.51}$	$\mathbf{88.76 \pm 0.73}$
	ResNet56	82.71 ± 0.55	86.84 ± 0.04	88.22 ± 0.41
	ResNet56 + AS	$\mathbf{87.30 \pm 0.32}$	$\mathbf{88.86 \pm 0.79}$	$\mathbf{89.61 \pm 0.93}$
	VGG	85.04 ± 0.44	88.47 ± 0.12	89.80 ± 0.12
	VGG + AS	$\mathbf{88.68 \pm 0.53}$	$\mathbf{90.39 \pm 0.35}$	$\mathbf{90.86 \pm 0.32}$
CIFAR-100	ResNet20	52.54 ± 1.54	55.58 ± 1.33	56.56 ± 1.07
	ResNet20 + AS	$\mathbf{54.83 \pm 0.95}$	$\mathbf{57.24 \pm 0.89}$	$\mathbf{57.62 \pm 0.91}$
	ResNet56	44.92 ± 5.58	51.67 ± 2.99	53.21 ± 2.40
	ResNet56 + AS	$\mathbf{54.76 \pm 2.26}$	$\mathbf{57.50 \pm 1.43}$	$\mathbf{58.63 \pm 1.58}$
	VGG	40.61 ± 1.28	45.38 ± 0.95	47.60 ± 1.01
	VGG + AS	$\mathbf{51.14 \pm 1.23}$	$\mathbf{54.95 \pm 0.53}$	$\mathbf{56.11 \pm 0.66}$

Table 1: **Comparison of models trained with Adversarial Sampling (AS) and models trained without Adversarial Sampling.** The lowest accuracy and highest accuracy among 100 sampled models, and the accuracy of the ensembled model are used as the evaluation metric. The mean value and maximum deviation of three runs are reported.

Dataset	Model	Lowest Accuracy	Highest Accuracy	Ensembled Accuracy
CIFAR-10	ResNet20	86.35 ± 0.62	90.29 ± 0.28	91.88 ± 0.06
	ResNet20 + AS	$\mathbf{88.19 \pm 0.44}$	$\mathbf{91.22 \pm 0.18}$	$\mathbf{91.98 \pm 0.18}$
	ResNet56	85.54 ± 1.24	90.48 ± 0.51	92.34 ± 0.44
	ResNet56 + AS	$\mathbf{88.74 \pm 0.64}$	$\mathbf{91.78 \pm 0.16}$	$\mathbf{92.75 \pm 0.25}$
	VGG	87.01 ± 1.04	90.23 ± 0.20	91.93 ± 0.26
	VGG + AS	$\mathbf{90.44 \pm 0.52}$	$\mathbf{91.92 \pm 0.06}$	$\mathbf{92.92 \pm 0.08}$
CIFAR-100	ResNet20	61.05 ± 0.61	64.53 ± 0.50	$\mathbf{66.97 \pm 0.72}$
	ResNet20 + AS	$\mathbf{63.51 \pm 0.64}$	$\mathbf{65.71 \pm 0.32}$	66.74 ± 0.77
	ResNet56	60.51 ± 1.30	64.99 ± 0.33	68.16 ± 0.12
	ResNet56 + AS	$\mathbf{64.93 \pm 0.43}$	$\mathbf{67.37 \pm 0.32}$	$\mathbf{69.48 \pm 0.39}$
	VGG	47.07 ± 2.35	52.00 ± 0.68	55.07 ± 1.05
	VGG + AS	$\mathbf{61.73 \pm 0.38}$	$\mathbf{64.18 \pm 0.67}$	$\mathbf{66.07 \pm 1.05}$

Table 2: **Comparison of models trained with Adversarial Sampling (AS) and models trained without Adversarial Sampling with Bayesian fine-tune.** The lowest accuracy and highest accuracy among 100 sampled models, and the accuracy of the ensembled model are used as the evaluation metric. The mean value and maximum deviation of three runs are reported.

works (Krishnan, Subedar, and Tickoo 2020; Deng et al. 2020). We implement Bayesian fine-tune as a higher baseline and test the impact of our Adversarial Sampling method on models trained with Bayesian fine-tune. The results are shown in Table 2. Similarly, the three kinds of accuracies are used as the evaluation metrics. Even though Bayesian fine-tune has improved the model performance to a large extent, models trained with the Adversarial Sampling method also perform obviously better compared with original models, proving the universality and efficacy of Adversarial Sampling.

5.5 The Ability of Uncertainty Estimation

To further verify that Bayesian neural networks trained with Adversarial Sampling still keep the ability to model the uncertainties, we measure both the aleatoric uncertainty and the epistemic uncertainty on our model with normal input images and blurred input images. The histograms for uncertainties obtained from models trained with Adversarial Sampling are plotted in Figure 6. It indicates that blurred images have larger uncertainty estimations on both aleatoric uncertainty and epistemic uncertainty. These findings show that the uncertainty estimates obtained from the model trained with the Adversarial Sampling method are useful and can identify the out-of-distribution data.

The uncertainties estimated by Bayesian neural networks serve as a measure of how confident the predictions are. Therefore, the predictions on which the model is least confident can be withdrawn, i.e., with higher uncertainty. Following previous papers (Filos et al. 2019; Krishnan, Subedar, and Tickoo 2020), we present the ensembled accuracies in Table 3 where only partial predictions are retained and the

Dataset	Model	20 % data retained	40 % data retained	60 % data retained	80 % data retained
CIFAR-10	ResNet20	99.82 ± 0.08	99.62 ± 0.14	98.55 ± 0.22	94.45 ± 0.30
	ResNet20 + AS	$\mathbf{99.90 \pm 0.05}$	$\mathbf{99.74 \pm 0.11}$	$\mathbf{99.07 \pm 0.15}$	$\mathbf{96.21 \pm 0.35}$
	ResNet56	99.90 ± 0.05	99.75 ± 0.10	98.81 ± 0.23	95.14 ± 0.61
	ResNet56 + AS	$\mathbf{99.95 \pm 0.00}$	$\mathbf{99.81 \pm 0.06}$	$\mathbf{99.21 \pm 0.11}$	$\mathbf{96.84 \pm 0.48}$
	VGG	99.88 ± 0.03	99.74 ± 0.09	99.28 ± 0.17	96.54 ± 0.10
	VGG + AS	$\mathbf{99.93 \pm 0.03}$	$\mathbf{99.79 \pm 0.09}$	$\mathbf{99.44 \pm 0.06}$	$\mathbf{97.68 \pm 0.34}$
CIFAR-100	ResNet20	96.40 ± 0.50	85.05 ± 1.45	74.09 ± 1.41	64.87 ± 1.26
	ResNet20 + AS	$\mathbf{96.68 \pm 0.68}$	$\mathbf{87.39 \pm 1.59}$	$\mathbf{76.43 \pm 1.29}$	$\mathbf{66.53 \pm 1.06}$
	ResNet56	93.88 ± 0.73	80.38 ± 1.29	69.82 ± 2.13	61.09 ± 2.56
	ResNet56 + AS	$\mathbf{97.18 \pm 0.43}$	$\mathbf{88.09 \pm 1.89}$	$\mathbf{77.20 \pm 1.85}$	$\mathbf{67.35 \pm 1.94}$
	VGG	86.93 ± 0.28	72.96 ± 0.36	62.96 ± 0.76	54.59 ± 0.82
	VGG + AS	$\mathbf{96.32 \pm 0.23}$	$\mathbf{85.61 \pm 0.59}$	$\mathbf{74.28 \pm 0.72}$	$\mathbf{64.79 \pm 0.73}$

Table 3: **Comparison of classification accuracy when only partial data are retained according to the uncertainties of the predictions.** The accuracies of the ensembled models from 100 samplings are used. The mean value and maximum deviation of three runs are reported.

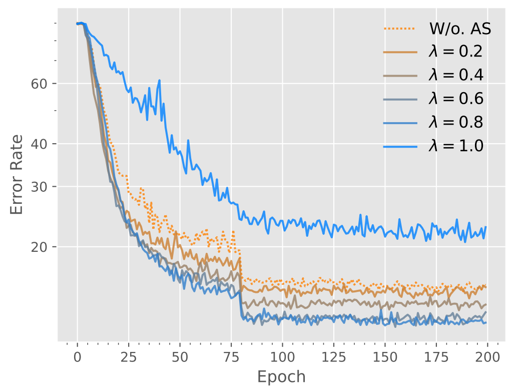


Figure 5: Error rates of models trained with different hyperparameter λ during a 200-epoch training. *Best viewed in color.*

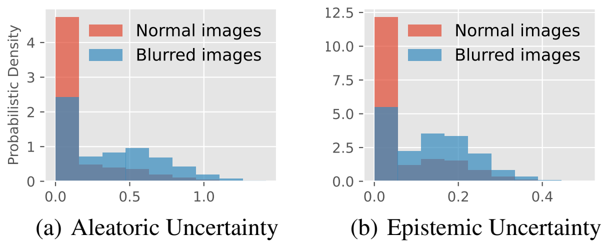


Figure 6: Uncertainties measured by Bayesian neural networks trained with Adversarial Sampling. Models trained with Adversarial Sampling keep the ability to model uncertainties. *Best viewed in color.*

rest are referred according to the total uncertainty. It can be seen as a more comprehensive evaluation metric on model performance and the ability to model uncertainties. The results show that the accuracy goes higher when fewer predictions are retained on all network structures and all datasets, which verifies the effectiveness of the estimated uncertainties. Models trained with the Adversarial Sampling have higher performance compared with models trained without Adversarial Sampling. Therefore, training with Adversarial Sampling approach is still helpful and enhances the model performance under this scenario, which confirms the effectiveness and versatility of the Adversarial Sampling method.

6 Conclusion

In this paper, we argue that the randomness of sampling in Bayesian neural networks causes the performance decrease. Then we propose to train Bayesian neural networks with Adversarial Distribution as a theoretical solution. Because the calculation of Adversarial Distribution is difficult in general, we further propose Adversarial Sampling as an approximation in practice. It tries to find the likely models with the worst performance by several iterative adversarial perturbations on the sampled parameters. Extensive experiments validate our proposal and show that models trained with Adversarial Sampling outperform models without Adversarial Sampling significantly. Moreover, models trained with the Adversarial Sampling method keep the ability to model the uncertainties, which further expands its application scenarios.

Acknowledgements

This work was supported in part by National NSF of China (NO. 61872234, 61732010), Shanghai Key Laboratory of Scalable Computing and Systems. Ruhui Ma is the corresponding author.

References

Antorán, J. 2019. *Understanding Uncertainty in Bayesian Neural Networks*. Master's thesis, University of Cambridge.

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein Generative Adversarial Networks. In *ICML*.

Bardenet, R.; Doucet, A.; and Holmes, C. 2017. On Markov Chain Monte Carlo Methods for Tall Data. *The Journal of Machine Learning Research*, 18(1): 1515–1557.

Blei, D. M.; Kucukelbir, A.; and McAuliffe, J. D. 2017. Variational Inference: A Review for Statisticians. *Journal of the American statistical Association*, 112(518): 859–877.

Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; and Wierstra, D. 2015. Weight uncertainty in neural network. In *ICML*.

Buntine, W. L. 1991. Bayesian Backpropagation. *Complex Systems*, 5: 603–643.

Carlini, N.; and Wagner, D. 2017. Towards Evaluating the Robustness of Neural Networks. In *SOSP*.

Chen, T.; Fox, E.; and Guestrin, C. 2014. Stochastic Gradient Hamiltonian Monte Carlo. In *ICML*.

Cheng, C.; Zhou, B.; Ma, G.; Wu, D.; and Yuan, Y. 2020. Wasserstein Distance based Deep Adversarial Transfer Learning for Intelligent Fault Diagnosis with Unlabeled or Insufficient Labeled Data. *Neurocomputing*, 409: 35–45.

Croce, F.; and Hein, M. 2020. Reliable Evaluation of Adversarial Robustness with an Ensemble of Diverse Parameter-Free Attacks. In *ICML*.

Deng, Z.; Zhang, H.; Yang, X.; Dong, Y.; and Zhu, J. 2020. BayesAdapter: Being Bayesian, Inexpensively and Reliably, via Bayesian Fine-Tuning. *arXiv preprint arXiv:2010.01979*.

Depeweg, S. 2019. *Modeling Epistemic and Aleatoric Uncertainty with Bayesian Neural Networks and Latent Variables*. Ph.D. thesis, Technische Universität München.

Depeweg, S.; Hernández-Lobato, J. M.; Doshi-Velez, F.; and Udluft, S. 2017. Learning and Policy Search in Stochastic Dynamical Systems with Bayesian Neural Networks. In *ICLR*.

Dong, Y.; Deng, Z.; Pang, T.; Zhu, J.; and Su, H. 2020. Adversarial Distributional Training for Robust Deep Learning. In *NeurIPS*.

Filos, A.; Farquhar, S.; Gomez, A. N.; Rudner, T. G.; Kenton, Z.; Smith, L.; Alizadeh, M.; de Kroon, A.; and Gal, Y. 2019. A Systematic Comparison of Bayesian Deep Learning Robustness in Diabetic Retinopathy Tasks. *arXiv preprint arXiv:1912.10481*.

Foti, N.; Xu, J.; Laird, D.; and Fox, E. 2014. Stochastic Variational Inference for Hidden Markov Models. In *NeurIPS*.

Frogner, C.; Zhang, C.; Mobahi, H.; Araya-Polo, M.; and Poggio, T. 2015. Learning with a Wasserstein Loss. In *NeurIPS*.

Gal, Y. 2016. *Uncertainty in Deep Learning*. Ph.D. thesis, University of Cambridge.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and Harnessing Adversarial Examples. In *ICLR*.

Graves, A. 2011. Practical Variational Inference for Neural Networks. In *NeurIPS*.

Gustafsson, F. K.; Danelljan, M.; and Schön, T. B. 2020. Evaluating Scalable Bayesian Deep Learning Methods for Robust Computer Vision. In *CVPR Workshops*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *CVPR*.

Hernández-Lobato, J. M.; and Adams, R. 2015. Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. In *ICML*.

Kendall, A.; and Gal, Y. 2017. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? In *NeurIPS*.

Kingma, D. P.; Salimans, T.; and Welling, M. 2015. Variational Dropout and the Local Reparameterization Trick. In *NeurIPS*.

Kingma, D. P.; and Welling, M. 2014. Auto-Encoding Variational Bayes. In *ICLR*.

Krishnan, R.; Subedar, M.; and Tickoo, O. 2020. Specifying Weight Priors in Bayesian Deep Neural Networks with Empirical Bayes. In *AAAI*.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images. *Tech Report*.

Kurakin, A.; Goodfellow, I.; and Bengio, S. 2017. Adversarial Examples in the Physical World. In *ICLR Workshop*.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *ICLR*.

Neal, R. M. 2012. *Bayesian Learning for Neural Networks*, volume 118.

Phan, B. T. 2019. *Bayesian Deep Learning and Uncertainty in Computer Vision*. Master's thesis, University of Waterloo.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*.

Smith, L.; and Gal, Y. 2018. Understanding Measures of Uncertainty for Adversarial Example Detection. In *UAI*.

Sun, L.; Tan, M.; and Zhou, Z. 2018. A Survey of Practical Adversarial Example Attacks. *Cybersecurity*, 1(1): 1–9.

Wenzel, F.; Roth, K.; Veeling, B.; Swiatkowski, J.; Tran, L.; Mandt, S.; Snoek, J.; Salimans, T.; Jenatton, R.; and Nowozin, S. 2020. How Good is the Bayes Posterior in Deep Neural Networks Really? In *ICML*.

Wu, D.; Xia, S.-T.; and Wang, Y. 2020. Adversarial Weight Perturbation Helps Robust Generalization. In *NeurIPS*.

Xiao, Y.; and Wang, W. Y. 2019. Quantifying Uncertainties in Natural Language Processing Tasks. In *AAAI*.

Zhang, J.; Hua, Y.; Xue, Z.; Song, T.; Zheng, C.; Ma, R.; and Guan, H. 2021. Robust Bayesian Neural Networks by Spectral Expectation Bound Regularization. In *CVPR*.

Zheng, Y.; Zhang, R.; and Mao, Y. 2021. Regularizing Neural Networks via Adversarial Model Perturbation. In *CVPR*.