Robust Bayesian Neural Networks by Spectral Expectation Bound Regularization

Jiaru Zhang\(^1\) Yang Hua\(^2\) Zhenguixue\(^1\) Tao Song\(^*\) Chengyu Zheng\(^1\) Ruhui Ma\(^1\) Haibing Guan\(^1\)†

\(^1\)Shanghai Jiao Tong University \(^2\)Queen’s University Belfast

{jiaruzhang, zhenguixue, songt333, zhengcy, ruhuima, hbguan}@sjtu.edu.cn, Y.Hua@qub.ac.uk

Abstract

Bayesian neural networks have been widely used in many applications because of the distinctive probabilistic representation framework. Even though Bayesian neural networks have been found more robust to adversarial attacks compared with vanilla neural networks, their ability to deal with adversarial noises in practice is still limited. In this paper, we propose Spectral Expectation Bound Regularization (SEBR) to enhance the robustness of Bayesian neural networks. Our theoretical analysis reveals that training with SEBR improves the robustness to adversarial noises. We also prove that training with SEBR can reduce the epistemic uncertainty of the model and hence it can make the model more confident with the predictions, which verifies the robustness of the model from another point of view. Experiments on multiple Bayesian neural network structures and different adversarial attacks validate the correctness of the theoretical findings and the effectiveness of the proposed approach.

1. Introduction

Bayesian neural networks [8, 29] provide a probabilistic view of deep learning frameworks by treating the model weights as random variables. One of the profound advantages of a Bayesian neural network is that it can provide both the aleatoric uncertainty and the epistemic uncertainty estimations because of the probabilistic representation of the model. In contrast, a vanilla deep neural network only models the aleatoric uncertainty by a certain probability distribution. Thus, Bayesian neural networks are successfully applied in many tasks to model uncertainties and build a more reliable and robust system, including but not limited to computer vision tasks [17, 20, 30] and natural language processing tasks [39].

Neural network models without particular settings [2, 14] are sensitive and vulnerable to adversarial attacks in testing. Defenses against adversarial attacks are difficult. The Lipschitz constant serves as an evaluation metric of the adversarial robustness of a model by providing a worst-case bound [18, 37]. Many previous methods enhance the model robustness by constraining the Lipschitz constant [10, 23, 31]. These methods have made a significant improvement in both theoretical analysis and practical applications. However, they cannot be used in Bayesian neural networks directly because of the probabilistic representations of model parameters.

Bayesian neural networks, on the other hand, are useful for defending adversarial noises compared with vanilla neural networks. Because of the probabilistic representations of model parameters and predictions, Bayesian neural networks can be applied to detect adversarial samples from normal samples [5, 24, 34]. Moreover, Bayesian neural networks have been found to have adversarial robustness naturally. Y. Gal et al. [13] and Carbone et al. [9] reveal that any gradient-based adversarial attacks are invalid on Bayesian neural networks under some extremely idealized conditions, e.g., idealized architecture [13], sufficient data and sampling times [9]. Nonetheless, these studies all have certain limitations. In many practical scenarios, predictions on adversarial samples are still necessary even though they have been detected. Additionally, the idealized conditions are almost impossible in practice. Therefore, there is still a vast space for further improvement of the robustness of Bayesian neural networks.

This paper presents a method, Spectral Expectation Bound Regularization (SEBR), to enhance the robustness of Bayesian neural networks. The model trained with SEBR has a smaller expectation of the spectral norm of the training parameter matrices. As a result, the improvement on the adversarial robustness of Bayesian neural networks is guaranteed based on theoretical derivation in this paper. Moreover, the impact of SEBR on the epistemic uncertainty of the output of Bayesian models is also studied theoretically and it further verifies the robustness of the proposed method. Ex-
shown that Bayesian neural networks are effective in defending against adversarial attacks in real tasks.

It is proved that SEBR training reduces the uncertainty of the model effectively in theoretical analysis, which provides another explanation of the model robustness.

Experiments on multiple Bayesian neural network structures verify the theory and the effectiveness of the proposed method. The codes are available in https://github.com/AISIGSJTU/SEBR.

2. Related Work

Robustness on Bayesian Neural Networks. Bayesian neural networks, where the model weights are treated as random variables, provide a probabilistic view of deep learning models [29]. Many previous methods investigated the robustness of Bayesian neural networks. It has been shown that Bayesian neural networks are effective in detecting adversarial samples [5, 24, 34], and it is observed that models tend to make wrong predictions on adversarial samples where the model outputs have high uncertainties [24]. X. Liu et al. applied adversarial training in Bayesian neural networks and gained an obvious robustness improvement [25]. From another point of view, Y. Gal et al. [13] revealed that idealized Bayesian neural networks can even avoid adversarial attacks. As the sufficient conditions in [13] are difficult to achieve in practice, Carbone et al. [9] further demonstrated that Bayesian neural networks are robust to gradient-based adversarial attacks in the large-data, over-parameterized limit. However, as the idealized conditions are almost impossible, Bayesian neural networks do not perform perfectly on defending against adversarial attacks in real tasks.

Lipschitz Constraint in Neural Networks. Methods about Lipschitz continuity are widely used to enhance the robustness and other targets in deep learning models. Yoshida and Miyato [40] proposed the spectral norm regularization to maintain the Lipschitz continuity by penalizing the sum of spectral norms of the parameter weight matrices. Further, Gouk et al. generalized the spectral norm regularization to non-$l_2$ norms and convolution layers [15]. On the other hand, Miyato et al. [28] proposed spectral normalization, where the spectral norms are normalized so that the Lipschitz constraint $Lip(f) = 1$ is satisfied. It is added into the discriminator in a generative adversarial network and the quality of generated samples gets improved. Jens Behrmann et al. [4] proved that the ResNet is invertible if its Lipschitz constant is restricted to $Lip(f) < 1$ on the residual blocks. Many other papers [3, 6, 12, 19, 32] apply the Lipschitz constraint and spectral norm in deep learning to enhance the generalizability and robustness. However, these existing methods are not suitable for Bayesian neural networks because of the probabilistic representations of parameters. Our method is the first to apply the Lipschitz constraint in Bayesian neural networks.

3. Background

3.1. Variational Inference in Bayesian Neural Networks

Suppose we have observations $D = \{(x_1, y_1), (x_2, y_2), \ldots \}$. A Bayesian neural network parameterized by $W$ uses a variational distribution $Q(W)$ to approximate the true posterior probability $P(W|D)$. For simplicity, we consider Bayesian neural networks with Gaussian priors, and parameters are represented as Gaussian distributions. The Bayesian neural network minimizes the Kullback–Leibler (KL) divergence

$$KL(Q(W)||P(W|D))) = - \int Q(W) \log \frac{P(W|D)}{Q(W)} dW$$

$$= \log(P(D)) - \int Q(W) \log \frac{P(W, D)}{Q(W)} dW.$$

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

$$L = - \int Q(W) \log \frac{P(W, D)}{Q(W)} dW$$

$$= -E_W \log P(D|W) + KL(P(W)||Q(W))).$$

Note that $-L$ is a lower bound of $\log P(X)$, thus $L$ is usually called the Evidence Lower Bound (ELBO) loss [7]. In practical Bayesian neural networks, the first term is usually estimated for each sample $(x, y)$ in the observations by the following Monte Carlo sampling

$$-E_W \log P(D|W) \approx - \frac{1}{K} \sum_{k=1}^{K} \log p(y|x, W_k), W_k \sim Q(W),$$

where $\log p(y|x, W_k)$ can be calculated by the cross-entropy loss in classification tasks. The second term $KL(P(W)||Q(W)))$ is directly computed analytically with a presumed prior distribution.
3.2. Lipschitz Continuity for Neural Networks

Lipschitz continuity is a significant property of a function in mathematical analysis. A function \( f : X \rightarrow Y \) is said to be Lipschitz continuous if there exists a real constant \( \alpha \geq 0 \) such that, for all \( x_1, x_2 \in X \), we have

\[
d_Y (f(x_1), f(x_2)) \leq \alpha \cdot d_X (x_1, x_2),
\]

where \( d_X \) and \( d_Y \) denote the distance metrics on set \( X \) and \( Y \), respectively. The smallest \( \alpha \) that satisfies this condition is referred to as the Lipschitz Constant of function \( f \). In the context of a deep neural network, the function \( f \) is a composite function composed by multiple functions:

\[
f(x) = (\phi_L \circ \phi_{L-1} \circ \cdots \circ \phi_1)(x),
\]

where each \( \phi_l \) is the mapping function of each layer \( l = 1, \cdots, L \). For the convenience of expression, we let \( Lip(\cdot) \) represent the Lipschitz constant of a function. According to the composition property of Lipschitz continuity, we have

\[
Lip(f) \leq \prod_{l=1}^{L} Lip(\phi_l).
\]

Hence, to constraint the Lipschitz constant of the whole function \( f \), it is sufficient to bound the Lipschitz constants for the mapping functions \( \phi_l \) of each layer \( l = 1, \cdots, L \).

3.3. Uncertainty Estimation

Uncertainty estimation is one of the significant functions of Bayesian neural networks, which is also essential in the evaluation of the robustness of a deep learning model [16, 24, 27]. For a classification model with parameters \( W \), input \( x \) and output \( y \) in classes \( C = \{c_1, c_2, \ldots, c_m\} \), following previous work [1, 11], we model the uncertainty in the prediction by its predictive entropy

\[
H(y|x, W) = \sum_{i=1}^{m} p(y = c_i|x, W) \log p(y = c_i|x, W).
\]

It contains both aleatoric uncertainty \( H_a \) and epistemic uncertainty \( H_e \). The aleatoric uncertainty \( H_a \) is given by

\[
H_a(y|x, W) = \mathbb{E}^W H(y|x, W) \approx \frac{1}{K} \sum_{k=1}^{K} H(y|x, W_k),
\]

which implies that it can be estimated by \( K \) Monte Carlo samplings. The epistemic component \( H_e \) is given by the difference between the total uncertainty \( H \) and the aleatoric uncertainty \( H_a \), i.e.,

\[
H_e(y|x, W) = H(y|x, W) - \mathbb{E}^W H(y|x, W).
\]

In a regression task, the output becomes a vector \( y \) instead of a class, which means the predictive entropy cannot be used to measure the uncertainty. Instead, the uncertainty can be measured by the following variance of the Gaussian mixture distribution over outputs [1]:

\[
H(y|x, W) = \sigma^2(y|x, W) = \frac{1}{K} \sum_{k=1}^{K} \sigma^2(y|x, W_k)
\]

where each \( W_k \) is the mixture distribution over outputs \( \sum_{k=1}^{K} \mu(y|x, W_k)^2 - \left( \frac{1}{K} \sum_{k=1}^{K} \mu(y|x, W_k) \right)^2 \).

The aleatoric uncertainty \( H_a \) is usually used to model the uncertainty caused by the noise in data, while the epistemic uncertainty \( H_e \) corresponds to the uncertainty in model parameters and model structures [1].

4. Spectral Expectation Bound Regularization

Here we consider an L-layer feed-forward Bayesian neural network to explain our method. A layer with the mapping function \( f_W(x) = f(Wx + b) \) accepts \( x \in \mathbb{R}^m \) as the input. Here \( f(\cdot) \) represents an activation function, e.g. relu and sigmoid, and \( W \) represents all trainable parameters of the function, including \( W \) and \( b \). The elements in parameter matrices \( W \) and \( b \) are all random variables in the Bayesian framework. Therefore, when we forward the function multiple times, the output vector \( y \) is sampled from a probabilistic distribution determined by input \( x \) and parameters \( W \).

In the following section, we will consider how to make the function robust to a given perturbation. The following theorem presents that the expectation of disturbance of the output in a layer is bounded by the expectation of the spectral norm of parameter matrix \( \mathbb{E}\|W\|_2 \), the length of the perturbation vector \( \|\xi\| \), and the Lipschitz constant of the activation function \( Lip(f) \).

**Theorem 1.** Consider function \( f_W(x) = f(Wx + b) \), where the activation function \( f(\cdot) \) is Lipschitz continuous with Lipschitz constant \( Lip(f) \). For any perturbation \( \xi \) with norm \( \|\xi\| \), we have

\[
\mathbb{E}\|f_W(x + \xi) - f_W(x)\| \leq Lip(f) \cdot \mathbb{E}\|W\|_2 \cdot \|\xi\|,
\]

where \( \|W\|_2 \) represents the spectral norm of matrix \( W \), and it is defined as

\[
\|W\|_2 = \max_{\xi \in \mathbb{R}^n, \xi \neq 0} \frac{\|W\xi\|}{\|\xi\|}.
\]
The proof of this theorem is given in the Supplementary Material. Note that the Lipschitz constant of the activation function \( f(\cdot) \) is fixed for a given Bayesian neural network structure. Besides, the Lipschitz constant of many popularly used activation function, e.g., relu and sigmoid, is bounded by 1. Therefore, the expectation of the spectral norm of the weight matrix can influence the sensitivity of a Bayesian neural network model. The model will become more robust if \( \mathbb{E}\|W\|_2 \) of each layer get restricted.

Similar to the spectral norm regularization in vanilla neural networks [40], a simple method to restrict \( \mathbb{E}\|W\|_2 \) in a Bayesian neural network model is to add it to the loss as a regularization term, i.e.,

\[
\minimize_{W} \mathcal{L} + \frac{\lambda}{2} \sum_{l=1}^{L} (\mathbb{E}\|W^l\|_2)^2, \tag{13}
\]

where the expectation is estimated by Monte Carlo sampling and the spectral norm is calculated by the Power Iteration method. However, this method has a very high computational complexity. We denote the times of Monte Carlo sampling as \( K \) and the iterations of Power Iteration as \( N \). Then, the time complexity of such calculation is \( O(KN) \). To accelerate the training process, we propose a method to fast estimate the upper bound of \( \mathbb{E}\|W\|_2 \) analytically, instead of directly estimating the expectation for \( W \) by Monte Carlo sampling and Power Iteration.

Theorem 2 gives an upper bound of the expectation of the spectral norm of a Gaussian random matrix.

**Theorem 2.** Consider a Gaussian random matrix \( W \in \mathbb{R}^{m \times n} \), where \( W_{ij} \sim N(M_{ij}, A_{ij}^2) \) with \( M, A \in \mathbb{R}^{m \times n} \). Suppose \( G \in \mathbb{R}^{m \times n} \) is a zero-mean Gaussian random matrix with the same variance, i.e., \( G_{ij} \sim N(0, A_{ij}^2) \). We have

\[
\mathbb{E}\|W\|_2 \leq \|M\|_2 + c \left( \max_i \|A_{i,:}\| + \max_j \|A_{:j}\| + \mathbb{E}\max_{i,j} |G_{ij}| \right), \tag{14}
\]

where \( c \) is a constant independent of \( W \).

The proof for this theorem is shown in the Supplementary Material. With Theorem 2, we do not need to directly optimize \( \mathbb{E}\|W\|_2 \). We can optimize the upper bound of \( \mathbb{E}\|W\|_2 \). Specifically, the Power Iteration method is utilized to estimate the spectral norm of \( \|M\|_2 \). Monte Carlo sampling is adopted to estimate the last term \( \mathbb{E}\max_{i,j} |G_{ij}| \). The remaining term, \( \max_i \|A_{i,:}\| + \max_j \|A_{:j}\| \) can be directly calculated on given \( A \). Therefore, the time complexity is reduced from \( O(KN) \) to \( O(K + N) \) successfully. The constant \( c \) can be simply ignored because it is independent of the input and our target is to minimize the whole algebraic expression. Therefore, we can add the upper bound as a regularisation term into the loss function. Consequently, we consider the following empirical risk minimization problem:

\[
\minimize_{W} \mathcal{L} + \lambda \cdot \mathcal{L}_S. \tag{15}
\]

Here \( \mathcal{L} \) is the ELBO loss as defined in Equation (2). The notation \( \mathcal{L}_S \) represents the SEBR loss:

\[
\mathcal{L}_S = \frac{1}{2} \sum_{l=1}^{L} (\|M^l\|_2 + \max_i \|A^l_{i,:}\| + \max_j \|A^l_{:j}\| + \sum_{k=1}^{K} \max_{i,j} |\epsilon_k \cdot A^l_{i,j}|^2), \epsilon_k \sim N(0, 1). \tag{16}
\]

The parameter \( \lambda \) is a regularization factor, which controls the trade-off between the robustness and the expressive power of the model. We refer to this method as Spectral Expectation Bound Regularization (SEBR). The algorithm to apply SEBR together with variational inference in practice is provided in Algorithm 1.

---

**Algorithm 1: Variational Inference with Spectral Expectation Bound Regularization**

1. Compute the ELBO loss \( \mathcal{L} \) with Equation (2).
2. \( \mathcal{L}_S = 0 \)
3. for \( l = 1 \) to \( L \) do
   4. Define \( M^l, A^l \) following Theorem 2.
   5. \( \mathcal{L}_l = 0 \)
   6. // First: \( \|M\|_2 \)
       sample \( u \sim N(0, 1) \)
   7. for Sufficient iterations \( N \) do
       8. \( v = (M^l)^T u / \|(M^l)^T u\| \)
       9. \( u = M^l v / \|M^l v\| \)
   10. end
   11. \( \mathcal{L}_l = \mathcal{L}_l + u^T M^l v \)
   12. // Second: \( \max_i \|A^l_{i,:}\| \) + \( \max_j \|A^l_{:j}\| \)
   13. \( \mathcal{L}_l = \mathcal{L}_l + \max_i \|A^l_{i,:}\| + \max_j \|A^l_{:j}\| \)
   14. // Third: \( \sum_{k=1}^{K} \max_{i,j} |\epsilon_k \cdot A^l_{i,j}|^2 \)
   15. \( \text{sum} = 0 \)
   16. for Sufficient MC simulation times \( K \) do
       17. sample \( \epsilon_k \sim N(0, 1) \)
       18. \( \text{sum} = \text{sum} + \max_{i,j} |\epsilon_k \cdot A^l_{i,j}| \)
   19. end
   20. \( \mathcal{L}_l = \mathcal{L}_l + \text{sum} / t \)
   21. \( \mathcal{L}_S = \mathcal{L}_S + \frac{1}{2} \mathcal{L}_l^2 \)
22. Update with the gradient on minimizing \( \mathcal{L} \).
Figure 1. The variation trends of both Monte Carlo estimation and the estimated upper bound of $E\|W\|_2$ in a 3-layer Bayesian neural network during training. *Best viewed in color.*

Figure 2. The comparison of the change of the Monte Carlo estimation and the estimated upper bound of $E\|W\|_2$ between the original model and the model trained with the SEBR method. *Best viewed in color.*

<table>
<thead>
<tr>
<th>Method</th>
<th>Avg. time per epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reg. on $E|W|_2$</td>
<td>1654.8 (s)</td>
</tr>
<tr>
<td>SEBR</td>
<td>410.5 (s)</td>
</tr>
</tbody>
</table>

Table 1. Time cost comparison between SEBR and the direct regularization on $E\|W\|_2$.

5. Influence of SEBR on Uncertainty

In this section, we show that our SEBR method can reduce the epistemic uncertainty on the output of a Bayesian neural network model.

The following theorem shows the epistemic uncertainty of the output of a one-layer Bayesian neural network decreases after one step gradient descent with SEBR.

**Theorem 3.** Consider a Bayesian neural network with only a linear layer $f_{W}(x) = Wx + b$, where $x \in \mathbb{R}^n$, $W \in \mathbb{R}^{m \times n}$. Denote the epistemic uncertainty (following the definition in Equation (10)) of the output after one step gradient descent without SEBR as $H_e$, and the epistemic uncertainty after one step gradient descent with SEBR as $H'_e$. With sufficient sampling times, we have

$$H'_e \leq H_e.$$  \hspace{1cm} (17)

**Proof.** With sufficient sampling times, the epistemic uncertainty the function $f_{W}(x) = Wx + b$ estimates is the variance of $\mu(y|x,W)$. Since $x$ is a constant vector and all elements of $W$ are independent Gaussian variables, we have

$$H_e = \sigma^2(\mu(y|x,W)) = \sigma^2\left(\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} W_{ij}x_j\right)$$

$$= \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{m} x_j^2 \sigma^2(W_{ij})$$

$$= \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{m} x_j^2 A^2_{ij},$$  \hspace{1cm} (18)

Here $A$ is the standard deviation matrix of $W$ following the definition in Theorem 2. Compared with normal training, the one step gradient descent with SEBR additionally optimize the SEBR loss $\mathcal{L}_S$. Let $A$ and $A'$ be the standard deviation matrices corresponding to training with SEBR and without SEBR respectively. For each $p = 1, 2, \ldots, m$ and $q = 1, 2, \ldots, n$,

$$A'_{pq} = A_{pq} - \alpha \sqrt{2\mathcal{L}_S} \left( \frac{\partial \|M\|_2}{\partial A_{pq}} + \frac{\partial \max_i \|A_{i,:}\|}{\partial A_{pq}} + \frac{\partial \max_j \|A_{:j}\|}{\partial A_{pq}} + \frac{\partial \sum_{k=1}^{K} \max_{i,j} \|\epsilon_k \cdot A_{ij}\|}{\partial A_{pq}} \right).$$  \hspace{1cm} (19)
where $\alpha > 0$ is the learning rate.

It is obvious that the first term $\frac{\partial \| M \|_2}{\partial A_{ij}} = 0$ since the mean matrix $M$ is unrelated to $A_{ij}$. The second term $\frac{\partial \max_p \| A_{ij} \|}{\partial A_{pq}} > 0$ when $p = \arg \max_i \| A_{ij} \|$; otherwise, $\frac{\partial \max_p \| A_{ij} \|}{\partial A_{pq}} = 0$. Similarly, the third term $\frac{\partial \max_i \| A_{ij} \|}{\partial A_{pq}}$ is non-negative. The last term satisfies

$$
\frac{\partial}{\partial A_{pq}} \sum_{k=1}^K \max_{i,j} |\epsilon_k \cdot A_{ij}| = \sum_{k=1}^K \frac{\partial \max_{i,j} |\epsilon_k \cdot A_{ij}|}{\partial A_{pq}} \geq 0.
$$

Therefore, we have: $\forall p, q, A'_{pq} \leq A_{pq}$. By substituting it into Equation (18), we can get the result of the theorem.

**6. Experiments**

In this section, we empirically verify our theoretical findings and investigate the effectiveness of the proposed SEBR method. We train a variety of Bayesian neural networks, including Bayesian MLPs and Bayesian CNNs, on MNIST dataset [22], Fashion-MNIST dataset [38], CIFAR-10 dataset, and CIFAR-100 dataset [21]. In Section 6.1,

<table>
<thead>
<tr>
<th>Model</th>
<th>Dataset</th>
<th>Attack</th>
<th>Noise</th>
<th>$\ell_{\infty}$ norm</th>
<th>Acc. w/o. SEBR (%)</th>
<th>Acc. w. SEBR (%)</th>
<th>$\Delta$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bayesian MLP</td>
<td>MNIST</td>
<td>FGSM</td>
<td>medium</td>
<td>0.16</td>
<td>8.97 ± 0.28</td>
<td>43.69 ± 5.92</td>
<td>+ 34.72</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>large</td>
<td>0.3</td>
<td>5.06 ± 0.21</td>
<td>24.54 ± 8.65</td>
<td>+ 19.48</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PGD</td>
<td>medium</td>
<td>0.16</td>
<td>4.20 ± 0.84</td>
<td>9.54 ± 2.82</td>
<td>+ 5.34</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>large</td>
<td>0.22</td>
<td>1.55 ± 0.35</td>
<td>3.18 ± 1.52</td>
<td>+ 1.63</td>
</tr>
<tr>
<td>Bayesian CNN</td>
<td>MNIST</td>
<td>FGSM</td>
<td>medium</td>
<td>0.08</td>
<td>55.98 ± 4.40</td>
<td>60.27 ± 8.65</td>
<td>+ 4.29</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>large</td>
<td>0.14</td>
<td>18.16 ± 0.57</td>
<td>22.55 ± 11.23</td>
<td>+ 4.39</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PGD</td>
<td>medium</td>
<td>0.08</td>
<td>36.53 ± 5.85</td>
<td>49.20 ± 10.75</td>
<td>+ 12.67</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>large</td>
<td>0.14</td>
<td>9.88 ± 2.02</td>
<td>12.33 ± 5.31</td>
<td>+ 2.45</td>
</tr>
<tr>
<td>Bayesian MLP</td>
<td>Fashion MNIST</td>
<td>FGSM</td>
<td>medium</td>
<td>0.1</td>
<td>24.29 ± 1.16</td>
<td>31.65 ± 1.25</td>
<td>+ 7.36</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>large</td>
<td>0.2</td>
<td>1.99 ± 0.57</td>
<td>4.59 ± 0.75</td>
<td>+ 2.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PGD</td>
<td>medium</td>
<td>0.1</td>
<td>19.18 ± 1.01</td>
<td>29.67 ± 1.22</td>
<td>+ 10.49</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>large</td>
<td>0.2</td>
<td>0.44 ± 0.14</td>
<td>2.71 ± 0.60</td>
<td>+ 2.27</td>
</tr>
<tr>
<td>Bayesian CNN</td>
<td>Fashion MNIST</td>
<td>FGSM</td>
<td>medium</td>
<td>0.08</td>
<td>15.89 ± 0.97</td>
<td>18.96 ± 5.00</td>
<td>+ 3.07</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>large</td>
<td>0.1</td>
<td>10.24 ± 0.31</td>
<td>11.97 ± 3.95</td>
<td>+ 1.73</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PGD</td>
<td>medium</td>
<td>0.06</td>
<td>15.03 ± 2.03</td>
<td>20.87 ± 4.00</td>
<td>+ 5.84</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>large</td>
<td>0.08</td>
<td>5.62 ± 0.73</td>
<td>9.27 ± 1.62</td>
<td>+ 3.65</td>
</tr>
</tbody>
</table>

Table 2. Comparison on the Robustness of Models without SEBR and with SEBR. The mean value and maximum deviation of three runs are reported.
Table 3. Comparison on the Robustness of Adversarial trained Models without SEBR and with SEBR. The mean value and maximum deviation of three runs are reported.

<table>
<thead>
<tr>
<th>Model</th>
<th>Dataset</th>
<th>Attack</th>
<th>Noise</th>
<th>$\ell_\infty$ norm</th>
<th>Acc. w/o. SEBR (%)</th>
<th>Acc. w. SEBR (%)</th>
<th>$\Delta$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bayesian MLP + Adv. Training</td>
<td>MNIST</td>
<td>FGSM</td>
<td>medium</td>
<td>0.16</td>
<td>54.56 ± 1.71</td>
<td>57.63 ± 1.08</td>
<td>+ 3.07</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9.94 ± 0.13</td>
<td>33.09 ± 8.23</td>
<td>+ 23.15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>small</td>
<td>0.04</td>
<td>92.57 ± 0.40</td>
<td>91.87 ± 0.26</td>
<td>−0.70</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PGD</td>
<td>medium</td>
<td>0.16</td>
<td>40.05 ± 5.32</td>
<td>40.66 ± 4.18</td>
<td>+ 0.61</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11.15 ± 5.70</td>
<td>16.47 ± 3.57</td>
<td>+ 5.32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>small</td>
<td>0.04</td>
<td>92.87 ± 0.27</td>
<td>92.08 ± 0.12</td>
<td>−0.79</td>
</tr>
</tbody>
</table>

6.1. Variation of Models after adding SEBR

We present the experiment results to verify that the upper bound of $E[\|W\|_2]$ is a suitable estimation and it can reflect the change trend of the real value of $E[\|W\|_2]$. The experiments are done on a Bayesian MLP with three layers on the MNIST dataset [22]. The values of $E[\|W\|_2]$ and our estimated upper bound are recorded for each layer during a 50-epoch training. Figure 1 shows the results. Even though there is an obvious gap between the upper bound and the unbiased estimated value by the Monte-Carlo estimation, the difference between them keeps stable and their variation trends are synchronous. This validates the rationality of utilizing the upper bound in our method.

We further investigate how our SEBR method influences $E[\|W\|_2]$. The parameter $\lambda$ is set to be 0.01 in the Bayesian neural network. The experiment results are shown in Figure 2. The added constraint on the upper bound from SEBR not only reduces the upper bound itself but also reduces the unbiased evaluated value estimated by the Monte Carlo estimation, which validates the effectiveness of SEBR.

To verify that SEBR indeed reduces the time cost in practice, we compare the time cost for SEBR and the direct regularization method on the expectation of the spectral norms shown in Equation (13). The simulation times of Monte Carlo sampling and iteration times of Power Iteration are set as 10. According to the experimental results shown in Table 1, the direct optimization on $E[\|W\|_2]$ makes the training very slow because it needs sufficient times for both Monte Carlo sampling and Power Iteration and the calculation of the expectation is necessary in every forward propagation. The training with SEBR significantly reduces the amount of time cost for training compared with the direct optimization method. Hence, it enhances the feasibility of the method in practice.

6.2. Improvements on Adversarial Robustness

The Fast Gradient Sign Method (FGSM) [14] is one of the most commonly used attack methods. The Projected Gradient Descent method (PGD) [26] is a more sophisticated and powerful adversarial attack method. To evaluate the impact of different settings of $\lambda$ for the SEBR method shown in Equation (15), we measure the change in robustness with varying $\lambda$ on defending the FGSM and the PGD attacks. The results are presented in Figure 3, where the accuracy is used as the evaluation metric. In the absence of adversarial noise, our SEBR causes a slight decrease on performance. It is normal because of the trade-off between clean accuracy and adversarial accuracy [35, 41]. With the increase of $\lambda$ from 0 (i.e., model without SEBR) to 0.02, the model becomes more robust on defending noises, even though there is a subtle performance decrease on data without adversarial noise. On the other hand, when we continue increasing $\lambda$, the model performs worse because of the poorer fitting ability. Therefore, using a suitable $\lambda$ is important to achieve fairly competent performance.

Table 2 provides the comparisons of robustness of the
models without SEBR and with SEBR, where both the Bayesian MLP models and the Bayesian CNN models are tested on the MNIST [22] dataset and the Fashion MNIST dataset [38]. We continue using the 3-layer neural network in the MLP model, and we use LeNet as the CNN architecture here. The hyper-parameter settings and the implementation details are reported in the Supplementary Material. We present the accuracy of the models on defending adversarial attacks of different norms. Since different adversarial attacks are not of the same attack power and the robustness of different baseline models are also different, different absolute noise $\ell_{\infty}$ norms are adopted for different models to reflect the robustness of the model in various situations as fully as possible. The models with SEBR are more robust on defending all of small, medium, large noises compared with the original Bayesian neural network models. To verify that SEBR is also effective on more modern architectures and larger datasets, we show more experiment results about SEBR of Bayesian CNN with VGG [33] architecture on CIFAR10 and CIFAR100 datasets in the Supplementary Material. SEBR keeps effective on the larger diverse datasets and more complex network architecture.

We also implement the model adversarially trained with FGSM as a higher baseline. It utilizes the information from model gradients and input data, and hence it is among the most effective defense techniques [14, 25, 36]. The results are shown in Table 3. It makes models robust on defending data with small noise. Nonetheless, our SEBR method further improves the model robustness obviously on defending larger adversarial noise, which further verifies the universality and effectiveness of SEBR.

6.3. Uncertainty Variation

To further verify the robustness of the models trained with SEBR, we measure the aleatoric uncertainty and the epistemic uncertainty on Bayesian neural networks trained with SEBR and without SEBR. Figure 4 presents the measured uncertainties on data with small FGSM adversarial noises ($\ell_{\infty} = 0.1$). More experimental results on clean data and other noises can be found in the Supplementary Material. All of the experiments show that the models trained with SEBR has lower uncertainties, including both the aleatoric uncertainty and the epistemic uncertainty. Therefore, our SEBR makes the models more confident on the predictions and improves the robustness.

7. Conclusion

In this paper, we propose the SEBR method that restricts the expectation of the Lipschitz constant on Bayesian neural networks. The theoretical analysis demonstrates that SEBR improves the robustness of defending against adversarial noises. The relationship between SEBR training and the output uncertainty variation is also discussed. It is proved that SEBR reduces the uncertainty on the model outputs. We verify our proposals by experiments on both the Bayesian MLP model and the Bayesian CNN model in defending FGSM and PGD attacks. Further experiments validate that models trained with SEBR have lower uncertainties, which verifies the robustness from another side.
References

[34] Lewis Smith and Yarin Gal. Understanding measures of uncertainty for adversarial example detection. In Amir Globerson and Ricardo Silva, editors, UAI, 2018. 1, 2


