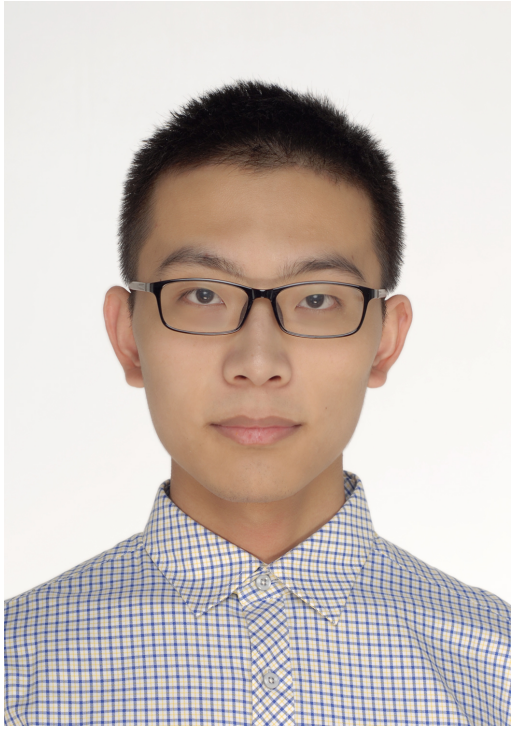
Robust Bayesian Neural Networks by Spectral Expectation Bound Regularization

Jiaru Zhang

Shanghai Key Laboratory of Scalable Computing and Systems

Bio

- Jiaru Zhang
- I am a Ph.D. candidate, major in Computer Science and Technology at SJTU.
- I am a member of the Shanghai Key Laboratory of Scalable Computing and Systems under the supervision of Prof. Haibing Guan.
- My research interest includes Bayesian Deep learning and Security in Deep learning.



Paper Overview

Robust Bayesian Neural Networks by Spectral Expectation Bound Regularization

Jiaru Zhang[1] Yang Hua[2] Zhengui Xue[1] Tao Song[1*] 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

This work has been accepted by CVPR 2021.

GitHub Repository: https://github.com/AISIGSJTU/SEBR

★ Search for **AISIGSJTU** and give us a star! Thanks!

Table of Contents

Background

Bayesian Neural Networks

- Bayesian neural network is a particular neural network where parameters are represented as **probabilistic distributions**.
- It is widely used because of its **uncertainty** and **interpretability**.

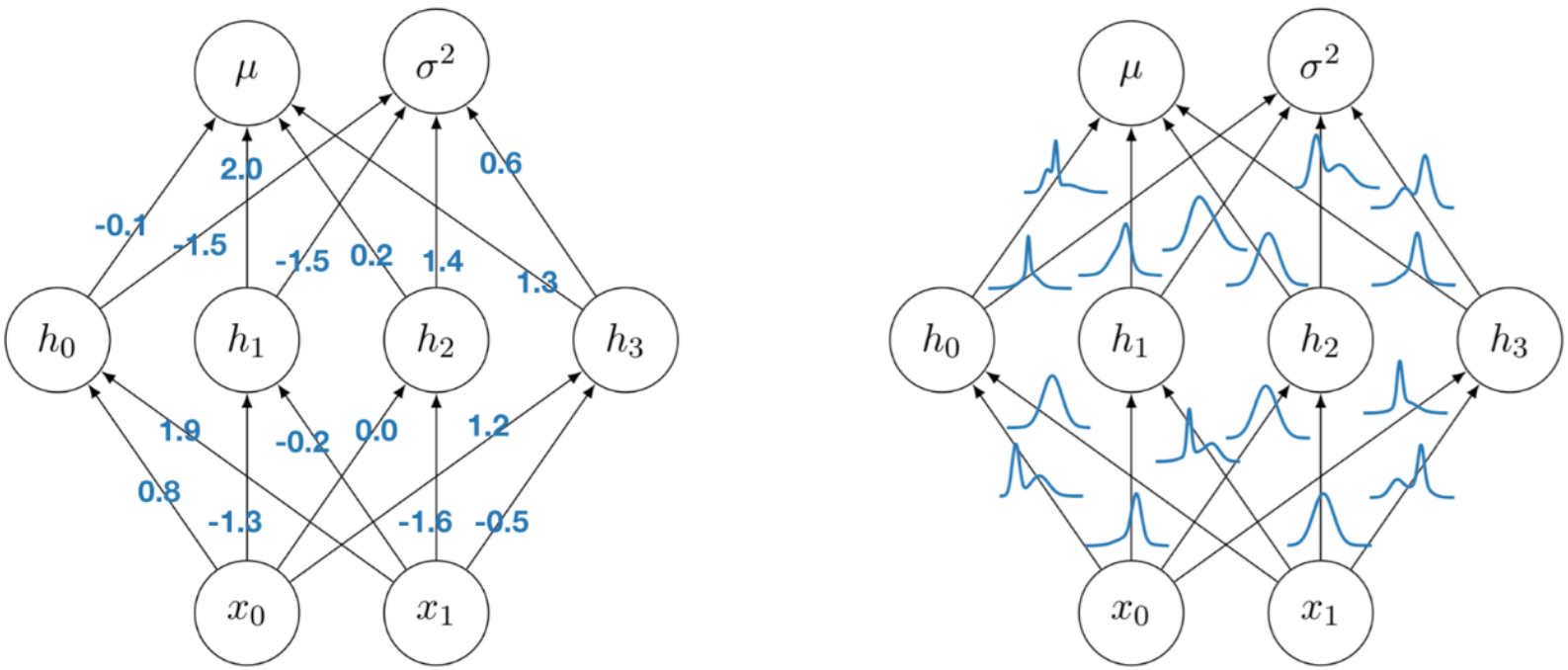


Figure source: Javier Antorán. Understanding uncertainty in bayesian neural networks. Masters thesis, University of Cambridge, 2019.

Uncertainties

- Consider a 3-classification problem, where the model outputs a probability for each of the classes.
- The vanilla neural network can model the **aleatoric uncertainty** H_a:
  - High H_a: [0.3, 0.3, 0.4]
  - Low H_a: [0.01, 0.01, 0.98]

Uncertainties

- Consider a 3-classification problem, where the model outputs a probability for each of the classes.
- The vanilla neural network can model the **aleatoric uncertainty** H_a:
  - High H_a: [0.3, 0.3, 0.4]
  - Low H_a: [0.01, 0.01, 0.98]

- A Bayesian neural network outputs different results in different runs. Hence, it can model the **epistemic uncertainty** H_e additionally :
  - High H_a and high H_e: [0.3, 0.3, 0.4], [0.2, 0.5, 0.3], [0.6, 0.2, 0.2], ...

Uncertainties

- Consider a 3-classification problem, where the model outputs a probability for each of the classes.
- The vanilla neural network can model the **aleatoric uncertainty** H_a:
  - High H_a: [0.3, 0.3, 0.4]
  - Low H_a: [0.01, 0.01, 0.98]

- A Bayesian neural network outputs different results in different runs. Hence, it can model the **epistemic uncertainty** H_e additionally :
  - High H_a and high H_e: [0.3, 0.3, 0.4], [0.2, 0.5, 0.3], [0.6, 0.2, 0.2], ...
  - High H_a but low H_e: [0.3, 0.3, 0.4], [0.3, 0.32, 0.38], [0.27, 0.3, 0.43], ...

Uncertainties

- Consider a 3-classification problem, where the model outputs a probability for each of the classes.
- The vanilla neural network can model the **aleatoric uncertainty** H_a:
  - High H_a: [0.3, 0.3, 0.4]
  - Low H_a: [0.01, 0.01, 0.98]

- A Bayesian neural network outputs different results in different runs. Hence, it can model the **epistemic uncertainty** H_e additionally :
  - High H_a and high H_e: [0.3, 0.3, 0.4], [0.2, 0.5, 0.3], [0.6, 0.2, 0.2], ...
  - High H_a but low H_e: [0.3, 0.3, 0.4], [0.3, 0.32, 0.38], [0.27, 0.3, 0.43], ...
  - Low H_a but high H_e: [0.01, 0.01, 0.98], [0.98, 0.01, 0.01], [0.01, 0.98, 0.01], ...

Uncertainties

- Consider a 3-classification problem, where the model outputs a probability for each of the classes.
- The vanilla neural network can model the **aleatoric uncertainty** H_a:
  - High H_a: [0.3, 0.3, 0.4]
  - Low H_a: [0.01, 0.01, 0.98]

- A Bayesian neural network outputs different results in different runs. Hence, it can model the **epistemic uncertainty** H_e additionally :
  - High H_a and high H_e: [0.3, 0.3, 0.4], [0.2, 0.5, 0.3], [0.6, 0.2, 0.2], ...
  - High H_a but low H_e: [0.3, 0.3, 0.4], [0.3, 0.32, 0.38], [0.27, 0.3, 0.43], ...
  - Low H_a but high H_e: [0.01, 0.01, 0.98], [0.98, 0.01, 0.01], [0.01, 0.98, 0.01], ...
  - Low H_a and low H_e: [0.01, 0.01, 0.98], [0.01, 0.02, 0.97], [0.02, 0.91, 0.97], ...

Examples of uncertainty modeling in CV

- Modeling both H_a and H_e gives a notable improvement in segmentation accuracy:

CamVid	IoU
SegNet [28]	46.4
FCN-8 [29]	57.0
DeepLab-LFOV [24]	61.6
Bayesian SegNet [22]	63.1
Dilation8 [30]	65.3
Dilation8 + FSO [31]	66.1
DenseNet [20]	66.9
This work:	
DenseNet (Our Implementation)	67.1
+ Aleatoric Uncertainty	67.4
+ Epistemic Uncertainty	67.2
+ Aleatoric & Epistemic	**67.5**

(a) CamVid dataset for road scene segmentation.

NYUv2 40-class	Accuracy	IoU
SegNet [28]	66.1	23.6
FCN-8 [29]	61.8	31.6
Bayesian SegNet [22]	68.0	32.4
Eigen and Fergus [32]	65.6	34.1
This work:		
DeepLabLargeFOV	70.1	36.5
+ Aleatoric Uncertainty	70.4	37.1
+ Epistemic Uncertainty	70.2	36.7
+ Aleatoric & Epistemic	**70.6**	**37.3**

(b) NYUv2 40-class dataset for indoor scenes.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In NeurIPS, 2017.

Examples of uncertainty modeling in NLP

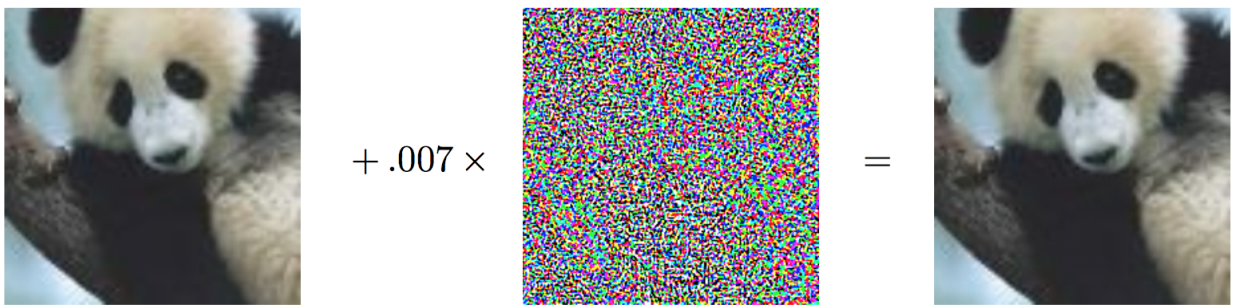
- Modeling both H_a (DU) and H_e (MU) also gives a notable improvement in sentiment analysis tasks:

Model	Yelp 2013	Yelp 2014	Yelp 2015	IMDB
(RGS MSE)				
Baseline	0.71	0.72	0.72	3.62
Baseline + MU	0.57	0.55	0.55	3.20
Baseline + DU	0.84	0.75	0.73	3.74
Baseline + both	**0.57**	**0.54**	**0.53**	**3.13**
Relative Improvement (%)	19.7	25.0	26.4	13.5

Yijun Xiao and William Yang Wang. Quantifying uncertainties in natural language processing tasks. In AAAI, 2019.

Adversarial Robustness

- Neural networks have been found **vulnerable to adversarial attacks**.
- A small perturbation which is undetectable for human can cause a large change of the output of a network.



- In this paper we focus on **adversarial defense**, i.e., how to defend the adversarial attacks.

Figure source: Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In ICLR, 2015

Adversarial Robustness on Bayesian Neural Networks

- Bayesian neural networks are effective in detecting adversarial samples [1] [2].
- Adversarial training has been used in Bayesian neural networks to improve model robustness [3].
- Idealized Bayesian neural networks can even avoid adversarial attacks under some settings [4] [5].
- However, there is still a large space for further improvement.

---

1 Yingzhen Li and Yarin Gal. Dropout inference in bayesian neural networks with alpha-divergences. In ICML, 2017.

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

3 Xuanqing Liu, Yao Li, Chongruo Wu, and Cho-Jui Hsieh. Adv-BNN: Improved adversarial defense through robust bayesian neural network. In ICLR, 2019.

4 Yarin Gal and Lewis Smith. Sufficient conditions for idealised models to have no adversarial examples: a theoretical and empirical study with bayesian neural networks. arXiv preprint arXiv:1806.00667, 2018.

5 Ginevra Carbone, Matthew Wicker, Luca Laurenti, Andrea Patane, Luca Bortolussi, and Guido Sanguinetti. Robustness of bayesian neural networks to gradient-based attacks. In NeurIPS, 2020.

Lipschitz Continuity

Consider a function $f(x)$ mapping x into y.

We hope it is insensitive to the perturbation of the input. For a perturbation ξ, we want

$$\|f(x+\xi) - f(x)\| \tag{1}$$

to be small. So we introduce the **Lipschitz Continuity** here:

Lipschitz Continuity

For a function f, if $\exists Lip(f)$, $\forall x, \xi$, we have

$$\|f(x+\xi) - f(x)\| \leq Lip(f) \cdot \|\xi\|, \tag{2}$$

then f is Lipschitz continuous or satisfies Lipschitz constraint.

Lipschitz Constraint in Neural Networks

- We have seen insensitivity means Lipschitz continuity. Additionally, we hope $Lip(f)$ as small as possible for a model f_w.
- Consider a single layer in a neural network:

$$f_w(x) = f(Wx + b), \tag{3}$$

  where W and b are parameter matrix and vector, $f(\cdot)$ is the activation function.

- If ξ is small enough,

$$\begin{aligned} \|f_w(x+\xi) - f_w(x)\| &= \|f(W(x+\xi)+b) - f(Wx+b)\| && (4)\\ &= \left\| \frac{\partial f}{\partial x} W\xi \right\| && (5)\\ &:\leq Lip(f) \cdot \|\xi\| && (6) \end{aligned}$$

- For popular activation functions (e.g. relu, sigmoid, tanh, ...), $\left\| \frac{\partial f}{\partial x} \right\|$ are all bounded.

Lipschitz Constraint in Neural Networks

- So we only need to maintain

$$\|W\xi\| \leq Lip(f) \cdot \|\xi\|, \tag{7}$$

  and answer the question: **What is the smallest Lip(f) ?**

- Now we introduce the definition of **Spectral Norm**:

Definition (Spectral Norm)

For a matrix W, we define its Spectral Norm as

$$\|W\|_2 = \max_{\xi \neq 0} \frac{\|W\xi\|}{\|\xi\|}. \tag{8}$$

Note that it is a generalization of l_2 norm for vectors.

- Now we can write the formula (7) as

$$\|W\xi\| \leq \|W\|_2 \cdot \|\xi\|. \tag{9}$$

Discussion

- The Lipschitz constraint is popularly used in deep learning to improvement the robustness and generality of models [6] [7] [8].
- However, it cannot be used in Bayesian neural networks because of the parameters in Bayesian neural networks are probabilistic distributions.

In a word, this work answers the following questions:

- **How to apply Lipschitz constraint in Bayesian neural networks to improve the adversarial robustness?**
- **How does the method influence the uncertainties?**

---

6 Yoshida, Yuichi and Miyato, Takeru. Spectral norm regularization for improving the generalizability of deep learning. arXiv preprint arXiv:1705.10941, 2017.

7 Gouk, H., Frank, E., Pfahringer, B., and Cree, M. Regularisation of neural networks by enforcing lipschitz continuity. arXiv preprint arXiv:1804.04368, 2018.

8 Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. In ICLR, 2018.

Spectral Expectation Bound Regularization

A Theoretical Point of Penetration

- The idea of restriction on $\|W\|_2$ can be naturally extended to Bayesian neural networks.
- It is proved that the model will become more robust if $\mathbb{E}\|W\|_2$ of each layer get restricted.

Theorem

Consider function $f_{\mathrm{W}}(\mathbf{x}) = f(W\mathbf{x} + \mathbf{b})$, *where the activation function* $f(\cdot)$ *is Lipschitz continuous with Lipschitz constant* $Lip(f)$. *For any perturbation* $\boldsymbol{\xi}$ *with norm* $\|\boldsymbol{\xi}\|$, *we have*

$$\mathbb{E}_{\mathrm{W}} \|f_{\mathrm{W}}(\mathbf{x}+\boldsymbol{\xi}) - f_{\mathrm{W}}(\mathbf{x})\| \leq Lip(f) \cdot \mathbb{E}\|W\|_2 \cdot \|\boldsymbol{\xi}\|, \tag{10}$$

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

$$\|W\|_2 = \max_{\boldsymbol{\xi}\in\mathbb{R}^n, \boldsymbol{\xi}\neq 0} \frac{\|W\boldsymbol{\xi}\|}{\|\boldsymbol{\xi}\|}. \tag{11}$$

Naive method to restrict $\mathbb{E}\|W\|_2$

How to restrict $\mathbb{E}\|W\|_2$ in practice? A naive method:

$$\underset{\mathrm{W}}{\text{minimize}} \quad \mathcal{L} + \frac{\lambda}{2} \sum_{l=1}^{L} (\mathbb{E}\|W^l\|_2)^2, \tag{12}$$

The expectation is estimated by **Monte Carlo sampling** (K times). The spectral norm is calculated by **Power Iteration** (N iterations) method.

The time complexity is $O(KN)$.

Faster Estimation

A substitution: Estimation of its **upper bound**.

Theorem

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*

$$\mathbb{E}\|W\|_2 \leq \|M\|_2 + c\left(\max_i \|A_{i,:}\| + \max_j \|A_{:,j}\| + \mathbb{E}\max_{i,j}|G_{ij}|\right), \quad (13)$$

where c *is a constant independent of* W.

The estimation of the upper bound is faster: $O(K + N)$

Denote $\mathcal{L}_S$ as half of the square of the upper bound of $\mathbb{E}\|W\|_2$ in each layer. Add it into the loss function:

$$\underset{W}{\text{minimize}}\ \mathcal{L} + \lambda \cdot \mathcal{L}_S. \quad (14)$$

The method is named as Spectral Expectation Bound Regularization (SEBR).

Verifications

- The upper bounds reflect the variation trends of real values accurately.

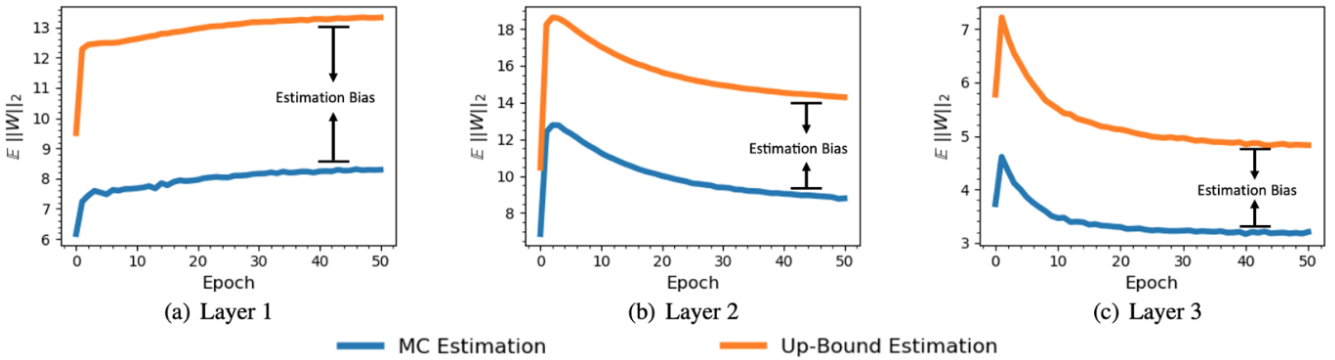


Figure 1. The variation trends of both Monte Carlo estimation and the estimated upper bound of $\mathbb{E}\|W\|_2$ in a 3-layer Bayesian neural network during training. *Best viewed in color.*

Verifications

- The upper bounds reflect the variation trends of real values accurately.

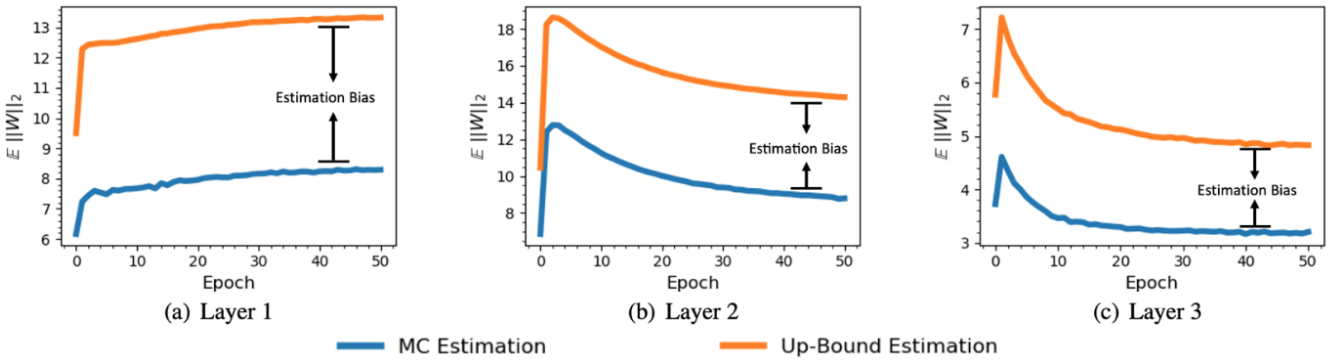
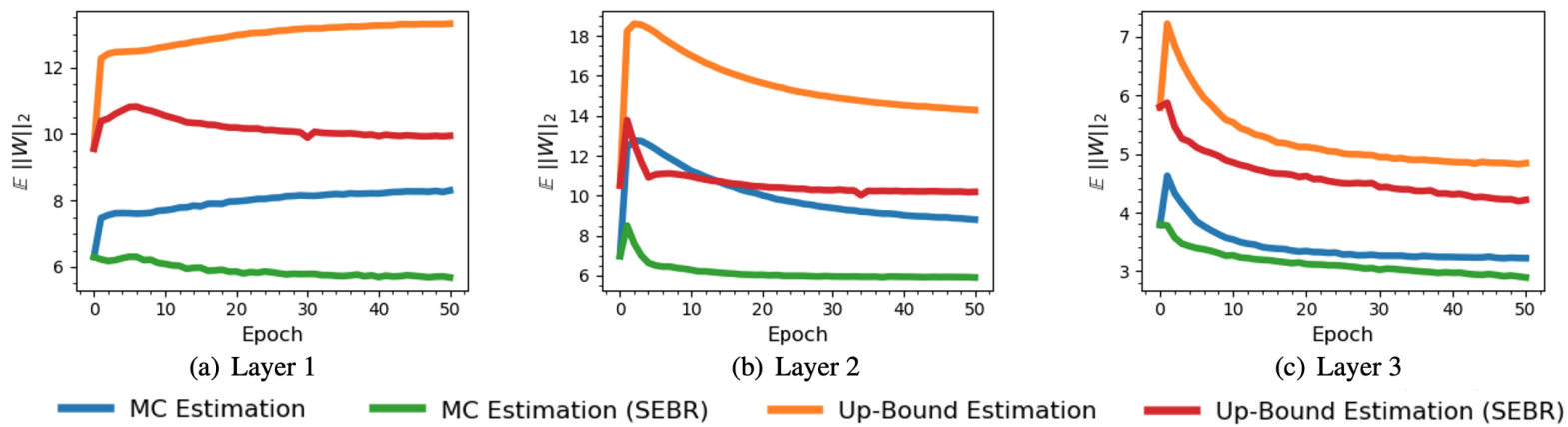


Figure 1. The variation trends of both Monte Carlo estimation and the estimated upper bound of $\mathbb{E}\|W\|_2$ in a 3-layer Bayesian neural network during training. *Best viewed in color.*

- The real values get decreased bacause of the usage of SEBR.



Verifications

- The time costs get reduced compared with the naive method.

Method	Avg. time per epoch
Reg. on $\mathbb{E}\|W\|_2$	1654.8 (s)
SEBR	410.5 (s)

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

Influence on Uncertainty

Influence on Uncertainty

The epistemic uncertainty of the model output gets reduced by SEBR:

Theorem

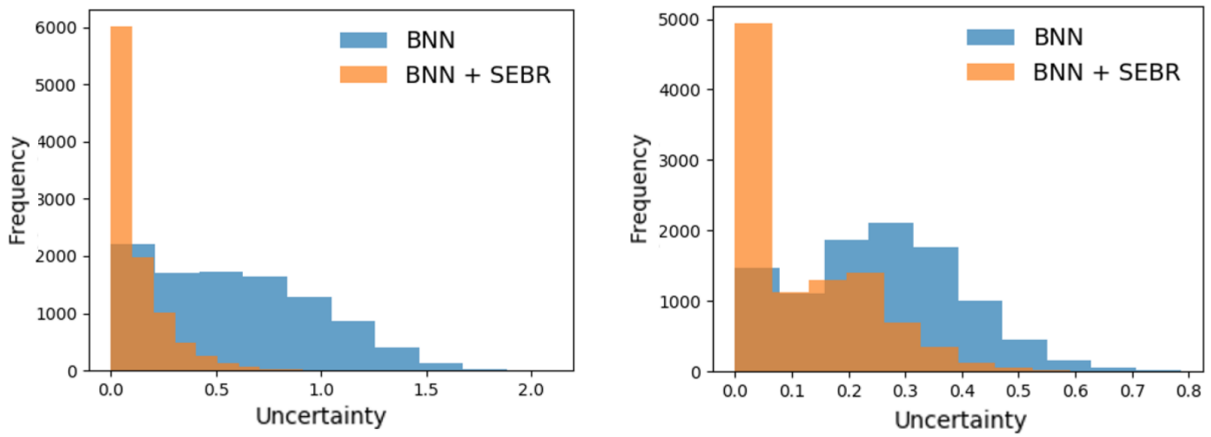
Consider a Bayesian neural network with only a linear layer $f_W(\mathbf{x}) = W\mathbf{x} + \mathbf{b}$, *where* $\mathbf{x} \in \mathbb{R}^n$, $W \in \mathbb{R}^{m \times n}$. *Denote the epistemic uncertainty of the output after one step gradient descent without SEBR as* H_e, *and the epistemic uncertainty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

$$H'_e \leq H_e. \tag{15}$$

It verifies the robustness of the model from another point of view.

Verification on Uncertainty Decrease

Experiments on the verification of the decrease of the output uncertainty.



(a) Aleatoric Uncertainty (b) Epistemic Uncertainty

Figure 4. Uncertainties measured by Bayesian neural networks on data with adversarial noises. Models trained with SEBR have lower uncertainty on the predictions. *Best viewed in color.*

Experiments of Improvement on Adversarial Robustness

Improvement on Adversarial Robustness

Experiments on multiple structures (i.e., MLP and CNN), multiple datasets (MNIST and Fashion-MNIST), and multiple attacks (i.e., FGSM and PGD) verify the efficiency of the proposed method.

Model	Dataset	Attack	Noise	ℓ_∞ norm	Acc. w/o. SEBR (%)	Acc. w. SEBR (%)	ΔAvg. Improv. (%)
Bayesian MLP	MNIST	/	0	0	97.05 ± 0.38	96.83 ± 0.48	-0.22
		FGSM	small	0.04	83.83 ± 0.51	85.74 ± 0.64	**+ 1.91**
			medium	0.16	8.97 ± 0.28	43.69 ± 5.92	**+ 34.72**
			large	0.3	5.06 ± 0.21	24.54 ± 8.65	**+ 19.48**
		PGD	small	0.04	81.99 ± 1.05	83.67 ± 0.67	**+ 1.68**
			medium	0.16	4.20 ± 0.84	9.54 ± 2.82	**+ 5.34**
			large	0.22	1.55 ± 0.35	3.18 ± 1.52	**+ 1.63**
Bayesian CNN	MNIST	/	0	0	98.88 ± 0.27	98.70 ± 0.04	-0.18
		FGSM	small	0.04	85.64 ± 2.52	86.14 ± 2.76	**+ 0.50**
			medium	0.08	55.98 ± 4.40	60.27 ± 8.65	**+ 4.29**
			large	0.14	18.16 ± 0.57	22.55 ± 11.23	**+ 4.39**
		PGD	small	0.04	82.91 ± 2.63	85.10 ± 2.96	**+ 2.19**
			medium	0.08	36.53 ± 5.85	49.20 ± 10.75	**+ 12.67**
			large	0.14	9.88 ± 2.02	12.33 ± 5.31	**+ 2.45**
Bayesian MLP	Fashion MNIST	/	0	0	84.38 ± 0.37	78.75 ± 0.83	-5.63
		FGSM	small	0.04	60.96 ± 0.24	62.06 ± 1.15	**+ 1.10**
			medium	0.1	24.29 ± 1.16	31.65 ± 1.25	**+ 7.36**
			large	0.2	1.99 ± 0.57	4.59 ± 0.75	**+ 2.60**
		PGD	small	0.04	59.86 ± 0.34	61.80 ± 1.13	**+ 1.94**
			medium	0.1	19.18 ± 1.01	29.67 ± 1.22	**+ 10.49**
			large	0.2	0.44 ± 0.14	2.71 ± 0.60	**+ 2.27**
Bayesian CNN	Fashion MNIST	/	0	0	87.45 ± 0.57	84.83 ± 0.33	-2.62
		FGSM	small	0.04	40.82 ± 1.86	46.03 ± 4.22	**+ 5.21**
			medium	0.08	15.89 ± 0.97	18.96 ± 5.00	**+ 3.07**
			large	0.1	10.24 ± 0.31	11.97 ± 3.95	**+ 1.73**
		PGD	small	0.04	32.81 ± 1.70	39.92 ± 3.25	**+ 7.11**
			medium	0.06	15.03 ± 2.03	20.87 ± 4.00	**+ 5.84**
			large	0.08	5.62 ± 0.73	9.27 ± 1.62	**+ 3.65**

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

Improvement on Adversarial Robustness

Experiments on more complex structures (i.e., VGG), more complex datasets (CIFAR-10/100), and multiple attacks (i.e., FGSM and PGD) further verify the efficiency of the proposed method.

Dataset	Attack	noise ℓ_∞	w/o. SEBR	w. SEBR
CIFAR10	/	0	91.65	**92.09**
	FGSM	0.005	58.65	**65.74**
		0.01	42.70	**54.78**
		0.02	32.73	**43.76**
	PGD	0.005	46.33	**50.40**
		0.01	9.73	**16.11**
		0.02	2.31	**2.95**
CIFAR100	/	0	**66.94**	66.56
	FGSM	0.002	45.96	**47.67**
		0.01	17.08	**21.18**
		0.02	12.52	**15.97**
	PGD	0.002	44.72	**46.85**
		0.01	2.91	**5.04**
		0.02	0.95	**1.95**

Table S1. Experiments on Bayesian CNN with VGG architecture.

Discussion

Discussion

There are some future directions about this work:

- Apply Lipschitz constraint on other kinds of neural networks.
- Apply the SEBR method in practical applications.
- Explore more methods to enhance the adversarial robustness of Bayesian neural networks.

Q&A