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

Introduction

- Bayesian neural networks have shown considerable potential and has been widely used in many tasks.
- Bayesian neural networks have indeed few publicized deployments in industrial practice despite the theoretical advancements.
- It is still unknown that why Bayesian neural networks can not learn a suitable representation and perform well.
- In this paper, we present a reason and propose Adversarial Sampling as a solution.

Background

- The learning target of Bayesian neural networks with variational inference is

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

- It can be divided into two terms.
- The first term $\mathcal{L}_p$ is directly related to the predictions.
- The second term $\mathcal{L}_r$ can be seen as a regularization on the model parameters.

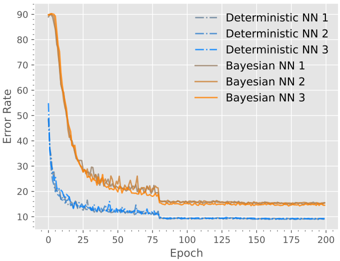
Explanation of the Poor Performance

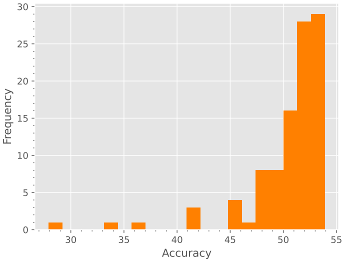
Because of the randomness of sampling during training and testing,

- There are some errors in updating the parameters.
- Some models with poor performance are yielded in random sampling.

Validation:

- The curves of Bayesian neural networks fluctuate more sharply.
- Some models have much lower accuracies compared with others.





Training with Adversarial Distribution

For dataset $\mathcal{D}$, parameter distributions $Q_\theta(\mathbf{W})$, we define

Adversarial Distribution

$$Q_{adv} = \underset{W[Q_{adv}, Q_\theta] \leq d}{\operatorname{argmax}} -\mathbb{E}_{\mathrm{W} \sim Q_{adv}(\mathrm{W})} \log P(\mathcal{D}|\mathbf{W}). \tag{1}$$

- $W[Q_{adv}, Q_\theta]$ denotes the Wasserstein distance between Q_{adv} and Q_θ.
- d is a hyperparameter to control $W[Q_{adv}, Q_\theta]$.

Training with Adversarial Distribution

Corresponding to the Adversarial Distribution, the adversarial loss is defined as

Adversarial Loss

$$\mathcal{L}_{adv} = -\mathbb{E}_{\mathrm{W} \sim Q_{adv}(\mathrm{W})} \log P(\mathcal{D}|\mathbf{W}) \tag{2}$$

The total learning target is

Total Learning Target

$$\theta = \underset{\theta}{\operatorname{argmin}} \left((1 - \lambda) \cdot \mathcal{L}_p + \lambda \cdot \mathcal{L}_{adv} + \mathcal{L}_r \right) \tag{3}$$

- λ controls the ratio of training with Adversarial Distribution.

Discussion

- The total learning target is equivalent to the original one when $d = 0$ or $\lambda = 0$.
- Sampling from the Adversarial Distribution yields likely models with the worst performance.
- Parameters updating accordingly guarantees the performance of the regularly sampled models.

Adversarial Sampling as an Approximation

- The calculation of Q_{adv} analytically is difficult.
- We propose an iterative approach, Adversarial Sampling, as an approximation.

Adversarial Sampling

1. Sample w_{adv} from the parameter distribution $N(\mu, \sigma^2)$.
2. Repeat multiple times:
   - $w_{adv} = w_{adv} + \alpha \cdot \sigma \cdot \mathbf{sign}\,(\mathbf{grad}\,(w_{adv}))$.

- Generating w_{adv} can be regarded as a sampling.
- Many $W_{adv}s$ create an approximation of Q_{adv}.

Implementation of Adversarial Sampling

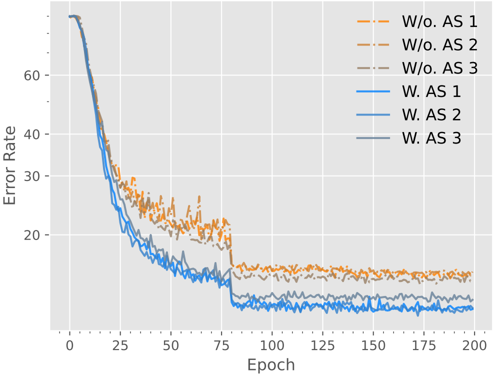
- With the reparameterization trick, w is from $w = \mu + \epsilon \cdot \sigma, \epsilon \sim \mathcal{N}(0, 1)$.
- It makes implementation easier and training with gradient desent possible by updating ϵ directly.

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)

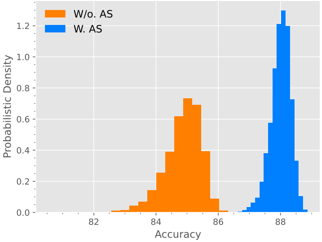
```
Sample ε_adv ~ N(0, 1)
for sufficient iterations N do
    Let w_adv = μ + ε_adv · σ
    Calculate the adversarial loss L_adv with parameter w_adv and data D
    Update ε_adv = ε_adv + α · sign(∂L_p / ∂ε_adv)
end for
Let w_adv = μ + ε_adv · σ
Calculate the adversarial loss L_adv with parameter w_adv and data D
Sample ε ~ N(0, I)
Let w = μ + ε · σ
Calculate the prediction loss L_p with parameter w and data D
Calculate the regularization loss L_r analytically
Calculate the total loss L with Equation (11)
Update parameter μ and σ with the total loss L
```

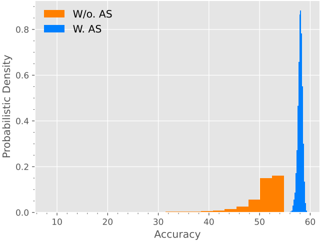
- Models trained with Adversarial Sampling have lower error rates.
- The change trends of models trained with Adversarial Sampling are more stable and steady.



Experiments: Verification of Motivation

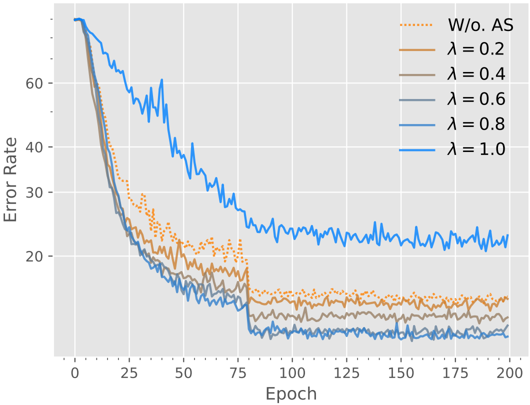
- The models trained without Adversarial Sampling distribute more dispersed.
- The accuracies of models trained with Adversarial Sampling are clearly higher.





Selection of Hyperparameter λ

- Model performance gets improved when λ increases from 0 to 0.8.
- There is a significant drop in performance when λ reaches 1.0.
- It validates the necessity of the introduction of parameter λ.



Improvement on Model Performance

We present three kinds of accuracies:

- The lowest accuracy and the highest accuracy among 100 sampled models.
- The accuracy of the ensembled model.

Models trained with Adversarial Sampling have much higher accuracies.

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	$\mathbf{51.14 \pm 1.23}$	$\mathbf{54.95 \pm 0.53}$	$\mathbf{56.11 \pm 0.66}$

Combination with Bayesian Fine-tune

- Bayesian fine-tune is an effective method to improve the performance.
- Models trained with Adversarial Sampling also perform better under this higher baseline.

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	$\mathbf{61.73 \pm 0.38}$	$\mathbf{64.18 \pm 0.67}$	$\mathbf{66.07 \pm 1.05}$

Uncertainty Estimation

- We present the ensembled accuracies where only partial predictions are retained according to the total uncertainty.
- Adversarial Sampling is still helpful under this scenario.

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	$\mathbf{96.32 \pm 0.23}$	$\mathbf{85.61 \pm 0.59}$	$\mathbf{74.28 \pm 0.72}$	$\mathbf{64.79 \pm 0.73}$

Conclusion

- We argue that the randomness of sampling in Bayesian neural networks causes the performance decrease.
- We propose training with Adversarial Distribution as a theoretical solution.
- We further propose Adversarial Sampling as an approximation in practice.
- Extensive experiments validate our proposal.

Thanks