Improving Neural Network Robustness with Bayesian Weight Sampling

Liyi Zhang, Yanda Chen

December 18, 2020

Abstract

Deep neural networks are the state-of-the-art for various tasks and benchmarks, but they are often not robust to slight or even negligible input perturbations. On image classification tasks, for example, models with near-perfect accuracy can easily degrade to near-zero accuracy when the input images change by a tiny amount that is invisible to human. We propose to improve models' robustness to input perturbations by adding diversity in model weights during training with Bayesian neural networks. Our experiments on an image classification benchmark shows that Bayesian neural networks are more robust than non-Bayesian deep neural networks trained with norm-based regularization. We claim that introducing diversity of model weights during model training improves model's robustness to input perturbations.

1 Introduction

Bayesian neural networks (BNN) view each weight of a neural network as a probability distribution. Early works on BNN by MacKay and Neal introduce prior distributions to neural network weights and use Markov Chain Monte Carlo methods for inference (**??**). Improving scalability, **?** introduces Bayes by Backprop, which uses Variational Inference and maximizes an approximate Evidence Lower Bound (ELBO). It uses gradient methods with the reparameterization trick (**??**) to maximize the ELBO by updating the trainable parameters to weights' variational distributions. By adding uncertainty to model weights, BNN achieves various advantages such as more accurate calibration of prediction confidence and more information-grounded model pruning (**?**).

In a parallel direction, previous research has shown that deep neural networks are not robust to adversarial input perturbations that are often tiny or negligible (**????**). Such perturbations are *universal* (we can find negligible perturbation for *each* input image to fool the model) and *generalizable across models* (adversarial perturbations targeting one model are generally hard for other models as well) (**?**). Multiple approaches have been proposed to improve model robustness to input perturbations, including sensitivity training (**?**), batch adjusted network gradients training (**?**) and semi-supervised training with unlabeled data (**?**). In this work, we propose to improve adversarial robustness of models by introducing weight diversity with Bayesian weight sampling.

2 Methods

2.1 Bayesian Neural Networks

We use a Gaussian variational posterior and a scale mixture prior for model weights of our Bayesian neural networks (**?**). The posterior distribution of each weight is parameterized with $(\mu, \rho) \in \mathbb{R}^2$ with probability density

$$p(w; \mu, \rho) = \mathcal{N}(w; \mu, \log(1 + \exp(\rho))) \tag{1}$$

We parameterize the posterior distribution in this way so that the variance parameter ρ is allowed to be negative. Both μ and ρ are trainable parameters updated during model training.

The prior distribution of each weight is parameterized with hyperparameters $(\sigma_1, \sigma_2, \pi) \in \mathbb{R}^3$ with probability density

$$p(w; \sigma_1, \sigma_2, \pi) = \pi \mathcal{N}(w; 0, \sigma_1^2) + (1 - \pi)\mathcal{N}(w; 0, \sigma_2^2) \tag{2}$$

We use the optimization procedure proposed by **?**:

1. Sample $\epsilon \sim \mathcal{N}(0, I)$.
2. Weight sampling: set $w = \mu + \log(1 + \exp(\rho)) \cdot \epsilon$.
3. Let $f(w, \mu, \rho) = \log q(w|\mu, \rho) - \log P(w)P(D|w)$
4. Update the variational parameters:

$$\mu \leftarrow \mu - \alpha \frac{df}{d\mu} \tag{3}$$

$$\rho \leftarrow \rho - \alpha \frac{df}{d\rho} \tag{4}$$

where α is the learning rate, and D is the set of training examples. During inference, we use the weight sampled from Step 2 for forward propagation. In other words, both training and inference of Bayesian neural networks are non-deterministic and randomized. We refer readers to **?** for more details on optimization and inference procedures.

2.2 Adversarial Perturbations

Let f denote a K-way classification model. Let (x, y) denote an input example with feature $x \in \mathbb{R}^d$ and label $y \in \mathbb{1}^K$. We say that $r \in \mathbb{R}^d$ is an *adversarial perturbation* if (a) $|r|$ is small, and (b) $f(x + r) \neq y$. We generate adversarial perturbation r for (f, x, y) by finding the r with minimum norm such that $f(x + r) \neq y$. Since it is hard to directly optimize this objective, we optimize a softer version of the objective following **?**:

$$r = \arg\min_{r \in \mathbb{R}^d} c|r| + l(f, x + r, y) \tag{5}$$

where $l(f, x + r, y)$ is the loss incurred by model f on the example $(x + r, y)$ and $c \in \mathbb{R}$ is a hyperparameter. Different from **?** who use a Limited-memory BFGS algorithm for optimization, we use a gradient descent optimization method to achieve even smaller $|r|$ norms compared to **?**.

3 Experimental Setup

3.1 Dataset

We use a standard 10-way image classification benchmark MNIST (**?**) for our experiments. We use the standard data split: 50k examples for training, 10k examples for validation and 10k examples for testing. Following **?**, we linearly scale input images to the interval (0, 2) as a pre-processing step.

3.2 Bayesian Neural Networks

Since the main focus of this work is not model architecture, we train simple fully connected networks FC-100-100-10 with 100 hidden units for both hidden layers and 10 units for the final output layer. We use Rectified Linear Units (ReLU) as activation for both hidden layers and SoftMax as activation for the final output layer. Following **?**, we use hyperparameters $\pi = 0.5, \sigma_1 = e^{-1}, \sigma_2 = e^{-7}$. We initialize μ with a Truncated Normal with the same mean and standard deviation as the weights' prior. We observe that the initialization of ρ is critical for Bayesian neural networks and controls the diversity and uncertainty of model weights. Therefore, we experimented with seven different constant initializations: $\rho \in \{-30, -10, -4, -3, -2.5, -2, -1.5\}$. For optimization, we use minibatches of size 128 and SGD optimizer with learning rate 0.001 and momentum 0.95. We use validation set accuracy for early stopping and set patience to be 5.

For fair comparison with non-Bayesian neural networks, we only use one sample of weights for forward propagation during inference. Since inference with one weight sampling is noisy, we average accuracies of 100 random runs. Note that this approach is different from approximate marginalization or ensemble learning, where one would take multiple samples of weights to form an averaged prediction and then compute a single accuracy.

3.3 Adversarial Perturbations

Adversarial perturbations are generalizable across models (**?**). In other words, examples modified with adversarial perturbations targeting one model are generally hard for other models as well (**?**). In order to generate a diverse collection of adversarial perturbations, we generate adversarial perturbations targeting three different non-Bayesian neural network architectures: FC-10, FC-200-200-10 and FC-1200-1200-10. To generate adversarial perturbations with different difficulty levels, for each model architecture we generate adversarial perturbations targeting four models trained with different L_2 regularization:

$$\mathscr{L} = \mathscr{L}_{\text{cross entropy}} + \lambda \sum_{l \in L} \frac{1}{|l|} \sum_{w \in l} w^2 \tag{6}$$

where L is the set of all layers and $|l|$ refers to the number of units in layer l. Because models trained with regularization are more robust to input perturbations, we conjecture that generated adversarial perturbations targeting models trained with large λ are intrinsically harder for other models as well. We experimented with four different λ values: 0 (no regularization), 10^{-4}, 10^{-2} and 1. Since we have three different model architectures and four different regularization coefficients, we generated 12 sets of adversarial perturbations. We generate adversarial perturbations for the 10k test examples.

Bayesian Neural Networks

$\log \rho_{\text{init}}$	-30.0	-10.0	-4.0	-3.0	-2.5	-2.0	-1.5
Validation	**97.51%**	97.28%	97.43%	97.29%	96.73%	96.25%	94.93%
Test	**97.51%**	97.46%	97.37%	97.12%	96.63%	96.07%	94.58%

Non-Bayesian Neural Networks

λ	0	10^{-4}	10^{-2}	1.0
Validation	97.57%	97.62%	**97.80%**	91.44%
Test	97.57%	97.64%	**97.94%**	90.91%

Table 1: Performance of Bayesian neural networks and non-Bayesian neural networks on unperturbed validation and test set. All neural networks have structure FC-100-100-10.

3.4 Baselines

We evaluate robustness of Bayesian neural networks and non-Bayesian neural networks by comparing their accuracy on the generated collection of adversarial perturbations. As a fair comparison, we train non-Bayesian neural networks of the same architecture as the Bayesian neural networks: FC-100-100-10. To create a stronger set of baselines, we train four models with different regularization coefficient λ: 0 (no regularization), 10^{-4}, 10^{-2} and 1. Allowing our baseline non-Bayesian models to use regularization leads to stronger baseline models that are intrinsically more robust.

4 Results & Analysis

4.1 Unperturbed Accuracy

We report the accuracy of Bayesian neural networks and non-Bayesian neural networks on unperturbed data in Table 1. We observe that for non-Bayesian neural networks, performance slightly improves when we use weak L_2 regularization ($\lambda = 10^{-2}$) but degrades sharply when we impose an overly strong regularization ($\lambda = 1$). For Bayesian neural networks, performance consistently decreases with increasing initialization of ρ. In other words, as we increase the weight uncertainty of Bayesian neural networks, performance consistently decreases. We conjecture that the reason behind this observation is that we are using only one weight sampling during inference and thus the inference is noisy and unstable. Overall, Bayesian neural networks slightly under-perform non-Bayesian neural networks on in-distribution unperturbed data.

4.2 Adversarial Perturbations

We generated adversarial perturbations following Section 3.3. We measure the average distortion of our generated adversarial perturbations as

$$\frac{1}{n}\sum_{i=1}^{n}\sqrt{\frac{|||r_i||_2^2}{d}}$$

Bayesian Neural Networks

Model Used to Generate Adversarial Perturbations	ρ_{init} of Evaluation Model						
	-30.0	-10.0	-4.0	-3.0	-2.5	-2.0	-1.5
FC-10 ($\lambda = 0$)	94.41%	94.54%	**95.14%**	94.89%	93.89%	92.54%	87.13%
FC-10 ($\lambda = 10^{-4}$)	93.36%	93.81%	**94.68%**	94.16%	93.16%	91.69%	85.49%
FC-10 ($\lambda = 10^{-2}$)	75.52%	75.13%	**77.44%**	76.80%	73.05%	69.56%	60.51%
FC-10 ($\lambda = 1$)	82.36%	82.09%	**83.48%**	83.23%	80.72%	77.70%	70.69%
FC-200-200-10 ($\lambda = 0$)	90.53%	90.12%	**91.41%**	90.91%	89.29%	88.05%	84.04%
FC-200-200-10 ($\lambda = 10^{-4}$)	92.47%	91.67%	**92.15%**	91.74%	90.15%	89.14%	85.41%
FC-200-200-10 ($\lambda = 10^{-2}$)	85.15%	85.05%	**86.43%**	86.31%	83.84%	82.20%	77.52%
FC-200-200-10 ($\lambda = 1$)	73.09%	72.65%	**74.39%**	74.22%	70.33%	66.51%	57.29%
FC-1200-1200-10 ($\lambda = 0$)	84.52%	82.33%	**86.57%**	86.27%	84.89%	82.92%	78.42%
FC-1200-1200-10 ($\lambda = 10^{-4}$)	83.07%	82.15%	85.82%	**86.25%**	84.42%	82.39%	78.02%
FC-1200-1200-10 ($\lambda = 10^{-2}$)	68.72%	69.85%	75.20%	**79.26%**	75.44%	72.45%	66.10%
FC-1200-1200-10 ($\lambda = 1$)	48.97%	48.08%	53.76%	**54.63%**	48.12%	43.15%	32.18%
Average	81.01%	80.62%	83.04%	**83.22%**	80.61%	78.19%	71.90%

Non-Bayesian Neural Networks

Model Used to Generate Adversarial Perturbations	λ of Evaluation Model			
	0	10^{-4}	10^{-2}	1
FC-10 ($\lambda = 0$)	90.07%	88.07%	**91.07%**	87.40%
FC-10 ($\lambda = 10^{-4}$)	88.68%	85.80%	**89.13%**	86.29%
FC-10 ($\lambda = 10^{-2}$)	77.85%	77.07%	**78.11%**	21.76%
FC-10 ($\lambda = 1$)	84.60%	84.51%	**84.96%**	34.38%
FC-200-200-10 ($\lambda = 0$)	78.43%	77.81%	**83.38%**	83.53%
FC-200-200-10 ($\lambda = 10^{-4}$)	82.73%	79.81%	**86.08%**	84.41%
FC-200-200-10 ($\lambda = 10^{-2}$)	48.58%	46.38%	49.74%	**83.28%**
FC-200-200-10 ($\lambda = 1$)	74.59%	74.59%	**75.49%**	7.77%
FC-1200-1200-10 ($\lambda = 0$)	49.52%	45.89%	53.79%	**82.49%**
FC-1200-1200-10 ($\lambda = 10^{-4}$)	49.19%	43.60%	51.43%	**82.64%**
FC-1200-1200-10 ($\lambda = 10^{-2}$)	20.61%	17.00%	15.94%	**79.95%**
FC-1200-1200-10 ($\lambda = 1$)	52.21%	50.74%	**54.28%**	10.03%
Average	66.42%	64.27%	**67.78%**	61.99%

Table 2: Performance of Bayesian and non-Bayesian neural networks on perturbed test set.

where r_i stands for the perturbation r for the i-th test example, n stands for the total number of test examples and d stands for the feature dimension 784. The average distortion of generated perturbations is < 0.10 for all 12 models, which is even smaller compared to **?**.

4.3 Robustness to Input Perturbations

We compare the robustness of Bayesian neural networks (Section 3.2) and non-Bayesian neural networks (Section 3.4) by comparing their performance on twelve sets of adversarial perturbations generated in Section 3.3. We show the results in Table 2.

First, we observe that for non-Bayesian neural networks, adding regularization does not consistently improve model robustness to input perturbations. In the only case where regularization does bring

Model Used to Generate Adversarial Perturbations	# of Ensembles of Evaluation Model				
	1	5	10	25	100
FC-10 ($\lambda = 0$)	94.75%	96.66%	96.85%	97.00%	**97.11%**
FC-10 ($\lambda = 10^{-4}$)	94.30%	96.22%	96.48%	**96.72%**	96.68%
FC-10 ($\lambda = 10^{-2}$)	77.02%	80.38%	80.53%	**81.25%**	81.22%
FC-10 ($\lambda = 1$)	83.44%	86.07%	86.22%	86.44%	**86.66%**
FC-200-200-10 ($\lambda = 0$)	90.97%	94.69%	95.34%	95.54%	**95.76%**
FC-200-200-10 ($\lambda = 10^{-4}$)	91.79%	95.07%	95.57%	95.94%	**96.13%**
FC-200-200-10 ($\lambda = 10^{-2}$)	86.60%	92.18%	93.04%	**93.76%**	93.64%
FC-200-200-10 ($\lambda = 1$)	74.35%	77.77%	**78.45%**	77.99%	78.18%
FC-1200-1200-10 ($\lambda = 0$)	86.59%	92.00%	92.67%	93.50%	**94.12%**
FC-1200-1200-10 ($\lambda = 10^{-4}$)	86.13%	91.59%	92.80%	93.37%	**93.48%**
FC-1200-1200-10 ($\lambda = 10^{-2}$)	78.28%	85.43%	87.31%	87.34%	**87.98%**
FC-1200-1200-10 ($\lambda = 1$)	54.28%	58.11%	**59.12%**	58.85%	58.41%
Average	83.21%	87.18%	87.86%	88.14%	**88.28%**

Table 3: Performance of Bayesian neural networks on perturbed test set when multiple weight samplings are used to form ensemble predictions.

improvement ($\lambda = 10^{-2}$), regularization only improves performance by 1.4 points.

Second, we observe that for non-Bayesian neural networks, performance on perturbed data improves with increasing ρ_{init}, reaches the maximum when $\rho_{\text{init}} = e^{-3}$, and then degrades when ρ_{init} further increases. This concave trend of performance versus ρ_{init} is very different from the monotone decreasing trend that we saw in Section 4.1 for performance on unperturbed data. Therefore, even though Bayesian weight uncertainty is not helpful on unperturbed data, weight uncertainty can improve model robustness to input perturbations.

Overall, we observe that Bayesian neural networks consistently out-perform non-Bayesian neural networks on all twelve sets of adversarial input perturbations by an average margin of **15.4 points**. It is also noteworthy that our Bayesian neural networks out-perform non-Bayesian neural networks with strong regularization ($\lambda = 1$) as well, which shows that Bayesian weight sampling also provides a very efficient regularization to improve model robustness.

5 Ensemble

Because each weight in a Bayesian neural network is a probability distribution, we can easily generate multiple sets of network weights by probabilistic sampling from weights distributions. We can use different sets of weights as ensemble during inference. Let N denote the number of weight samplings used for inference. Let $w_1, w_2, ..., w_N$ be network weights sampled from the trained Bayesian network weight distributions. Let $f(x|w_i)$ denote the prediction of feature x with network

Unperturbed Data						
Pruning Rate	25%	50%	75%	90%	95%	98%
Original Validation	97.47%	97.54%	95.63%	77.66%	54.21%	33.52%
Original Test	97.35%	97.49%	95.59%	76.62%	53.19%	31.98%
Perturbed Data						
Model Used to Generate	Pruning Rate					
Adversarial Perturbations	25%	50%	75%	90%	95%	98%
FC-10 ($\lambda = 0$)	95.36%	95.38%	90.54%	65.55%	46.11%	29.63%
FC-10 ($\lambda = 10^{-4}$)	94.83%	94.90%	89.63%	63.25%	45.51%	29.10%
FC-10 ($\lambda = 10^{-2}$)	77.96%	78.61%	72.23%	47.38%	33.67%	23.57%
FC-10 ($\lambda = 1$)	84.18%	84.32%	78.31%	54.46%	40.72%	29.67%
FC-200-200-10 ($\lambda = 0$)	91.80%	91.75%	84.34%	60.44%	43.50%	27.98%
FC-200-200-10 ($\lambda = 10^{-4}$)	92.74%	92.49%	85.19%	60.93%	43.52%	27.73%
FC-200-200-10 ($\lambda = 10^{-2}$)	87.87%	87.31%	78.15%	58.59%	43.97%	29.44%
FC-200-200-10 ($\lambda = 1$)	75.43%	76.08%	69.06%	44.80%	29.82%	21.32%
FC-1200-1200-10 ($\lambda = 0$)	87.69%	87.20%	77.95%	57.19%	42.31%	27.73%
FC-1200-1200-10 ($\lambda = 10^{-4}$)	87.43%	87.06%	76.96%	57.67%	42.55%	27.65%
FC-1200-1200-10 ($\lambda = 10^{-2}$)	80.21%	79.62%	66.12%	51.63%	43.14%	28.72%
FC-1200-1200-10 ($\lambda = 1$)	55.65%	56.01%	50.57%	30.83%	23.00%	18.40%
Average	84.26%	84.23%	76.59%	54.39%	39.82%	26.75%

Table 4: Performance of pruned Bayesian neural networks on unperturbed and perturbed data.

weight w_i. Using multiple weight samplings as ensemble, our final prediction is

$$f(x) = \frac{1}{N} \sum_{i=1}^{N} f(x|w_i) \tag{7}$$

All experiments in Section 4 use $N = 1$ for fair comparison with non-Bayesian neural networks. However, because inference with Bayesian neural networks is inherently noisy and unstable, we conjecture that a larger value of N may explore different areas of the loss function and produce more accurate, robust, and stable results. Therefore, we experimented with different values of N: 1, 5, 10, 25, 100 and show performance of ensemble Bayesian networks on perturbed data in Table 3. To report stable accuracy, for # of ensembles N, we report average accuracy across $100/N$ random runs. From Table 3 we can see that performance improves when we use ensemble predictions and increase the number of weight samplings during inference. When we use an ensemble of 100 weight samplings, the average performance across 12 sets of adversarial perturbations improves by 5.0 points compared to no ensemble. By using an ensemble of 100 weights, we improve average performance across 12 sets of adversarial perturbations by **20.5** points compared to the best-performing non-Bayesian neural networks in Table 2.

6 Model Pruning

Traditional model pruning techniques for non-Bayesian neural networks remove weights with small absolute values. However, for Bayesian neural networks where each weight is a probability distribution with trainable mean parameter μ and variance parameter σ, the signal-to-noise ratio $\frac{|\mu|}{\sigma}$ is more informative and leads to better performance when used for model pruning (**?**). Previous work has shown that performance of Bayesian neural networks on unperturbed in-distribution data is relatively stable under model pruning (**?**). Here we want to investigate whether performance of Bayesian neural networks on perturbed adversarial data is stable under model pruning as well.

We sort all network weights in increasing order of $\frac{|\mu|}{\sigma}$ and remove the weights with the smallest signal-to-noise-ratio. The pruning rate refers to the percentage of neural network weights removed. An ensemble of 100 weight samplings is used for this experiment. We show the performance of pruned Bayesian models on perturbed and unperturbed data with different pruning rates in Table 4.

From Table 4 we see that performance on unperturbed data stays the same when 50% of the network weights are removed. More importantly and somewhat surprisingly, performance on perturbed data also stays the same with 50% of weights pruned, which means that the robustness of Bayesian neural network is largely preserved even under significant model pruning. In fact, a Bayesian model pruned by 75% still out-performs the best-performing non-Bayesian model by a wide margin of **8.8 points** while being **75% smaller in size**.

7 Conclusion

In this work, we show that Bayesian weight sampling leads to neural networks that are significantly more robust to adversarial input perturbations compared to non-Bayesian neural networks. The diversity and uncertainty of network weights in Bayesian neural networks are able to mitigate the input perturbations and thus improve robustness. By exploiting other unique strengths of Bayesian weight sampling including ensembling and model pruning, we are able to build Bayesian neural network models that are significantly more robust and smaller in size compared to non-Bayesian neural network models.