Doctoral Dissertation

Bayesian Inference Approach for Robust Deep Neural Networks

KHONG THI THU THAO
Program of Information Science and Engineering
Graduate School of Science and Technology
Nara Institute of Science and Technology

Supervisor: Professor Yasuhiko Nakashima
Computing Architecture Lab. (Division of Information Science)

December 03, 2021

A Doctoral Dissertation
submitted to Graduate School of Science and Technology,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

KHONG THI THU THAO

Thesis Committee:

Supervisor	Prof. Yasuhiko Nakashima
	(Professor, Division of Information Science)
Co-supervisor	Prof. Kazushi Ikeda
	(Professor, Division of Information Science)
	Assoc. Prof. Renyuan Zhang
	(Associate Professor, Division of Information Science)
	Vis. Assoc. Prof. Tran Thi Hong
	(Visiting Associate Professor, Division of Information Science)

Bayesian Inference Approach for Robust Deep Neural Networks*

KHONG THI THU THAO

Abstract

The rapid deployment of deep neural networks (DNNs) and deep learning algorithms have been proving their enormous potentiality in a wide range of computer vision and the field of recognition. Nonetheless, due to a vulnerability, deep learning models' ability to complicated situations requires a fundamental tool for computer security. Recent studies have been shown a vulnerability of DNNs by a small adversarial perturbation in images that humans cannot distinguish and a well-trained neural network can misclassify. Therefore, there are many defense methods to improve the robustness of DNNs against adversarial attacks, for example, adversarial detection, statistical properties of network parameters, the normalization of input data, adversarial training, etc. Among them, adversarial training is an outstanding defense, but it is a challenge with respect to real data and large DNNs.

In order to avoid adversarial training, we have proposed a defense algorithm named Bayes without Bayesian Learning (BwoBL). Our algorithm builds Bayesian Neural Networks (BNNs) based on pre-trained DNNs and focuses on Bayesian inference without costing Bayesian learning. The stochastic components of BNNs can prevent the forceful gradient-based attacks and generate the ensemble model to enhance the DNN performance. As an application of transfer learning, BwoBL can easily integrate into any pre-trained DNN, which is trained on both natural and adversarial data. We have investigated the application of BwoBL to a variety of DNN architectures, such as Convolutional Neural Networks (CNNs) and Self-Attention Networks (SANs). It is believed that, unless

*Doctoral Dissertation, Graduate School of Science and Technology, Nara Institute of Science and Technology, December 03, 2021.

making DNN models larger, DNNs would be hard to strengthen the robustness to adversarial images. Our algorithm then employs scaling networks of CNNs and SANs, e.g., ResNet, EfficientNet, and SAN19 to construct BNNs against a diversity of adversarial attacks.

We assess the robustness of our BNN models by the top-1 accuracy on small datasets, i.e., CIFAR-10 and CIFAR-100, and the top-5 accuracy on real datasets like ImageNet. Our experiments utilize the currently strong attacks such as Projected Gradient Descent (PGD) and Carlini & Wagner (C&W) to produce adversarial examples. Experimental results have proved the efficiency of our BwoBL algorithm for resisting adversarial perturbation and solving the challenges of adversarial training and Bayesian learning.

Keywords:

Bayesian Neural Network (BNN), Deep Neural Network (DNN), Image Classification, Adversarial Attacks, Adversarial Robustness

Contents

List of Figures

List of Tables

1 Introduction

This part represents the overview of deep learning security, adversarial attacks of deep neural networks, and defense methods, which this dissertation has addressed. Then, the research contribution has been shown. The rest of this part presents the dissertation layout.

1.1 Overview

Owing to the wide applicability and super-human ability, machine learning systems have been employed in modern society as a general tool for computer applications. It is known that machine learning is an algorithm set to learn, executed on computer systems, and solves a process of information mining, pattern discovery, and inference from data [6]. When the algorithm learns, it alters itself based on data; as humans learn to make better decisions based on experience over time. If the brain controls all functions of a human, a neural network that is modeled like a human brain implements all machine learning algorithms. A neural network is designed to recognize the patterns of real-world data, such as images, sounds, text, etc., throughout the learning process or training.

The application of machine learning to various complex tasks, e.g., image classification [7, 8, 9], object detection [10, 11, 12], and natural language processing [13, 14] requires the use of multi-layer neural networks called Deep Neural Networks (DNNs), which are the stacking layers of statistical components to learn representations of data. Representation learning methods are deep learning that refers to a specific subset of machine learning [1, 15, 16]. Deep learning methods are obtained by composing computational modules of DNNs, in which each transforms the representation at the low level that starts with raw data, into the representation at the higher level. In image classification, for example, an image is formed by an array of pixel values, and the stacking layers of DNNs compute on these pixel values to capture and extract the features of the image, and the final layer combines the extracted features to detect the object in the image, as seen in Fig. 1. It is highlighted that the feature layers of DNNs are learned from given data, called a training set, without a design of humans. Compared to conventional machine learning models, DNNs could support humans to fea-

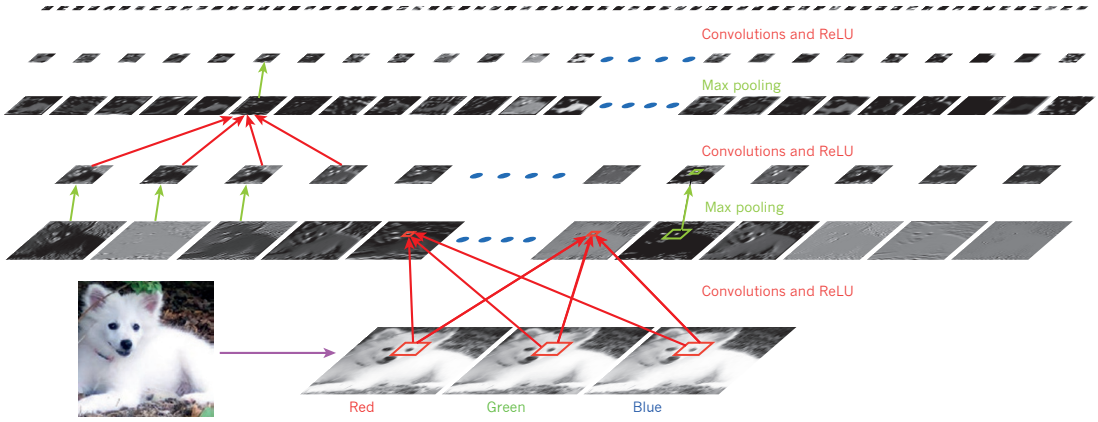

Figure 1. The feature extraction of a convolutional neural network in image classification [1].

ture techniques and yield better performance on a variety of visual benchmarks [17, 18, 19, 20]. Nevertheless, statistical analysis is a core of deep learning or machine learning models, which their outputs are often forms of probabilities and confidence intervals. Because of statistical properties, DNNs always exist a certain degree of risk that is a key aspect of deep learning security. DNNs are often designed without security and are vulnerable to a motivated adversary [6]. Therefore, an awareness of threat models must be maintained when building deep learning systems for security-critical applications such as self-driving cars and medical imaging.

DNNs have been shown to be vulnerable to adversarial perturbation [21, 22, 23, 24, 25, 26, 27, 28, 29] in the area of image recognition, which is crafted by adding a little noise to original images. This perturbation is imperceptible to human eyes but causes the misclassification of networks, as shown in Fig. 2. The lack of robustness for DNNs with respect to adversarial examples exposes security threats on the real systems of safety-critical applications that require high confidence. It is noted that adversaries can take the benefits of DNNs to avoid detection and defense. Defenders can learn from the attack methods and adjust themselves against attacks, and attackers can also explore the properties of defense approaches to craft a complete perturbation. Several attack methods have recently been developed to generate adversarial images, such as Fast Gradient

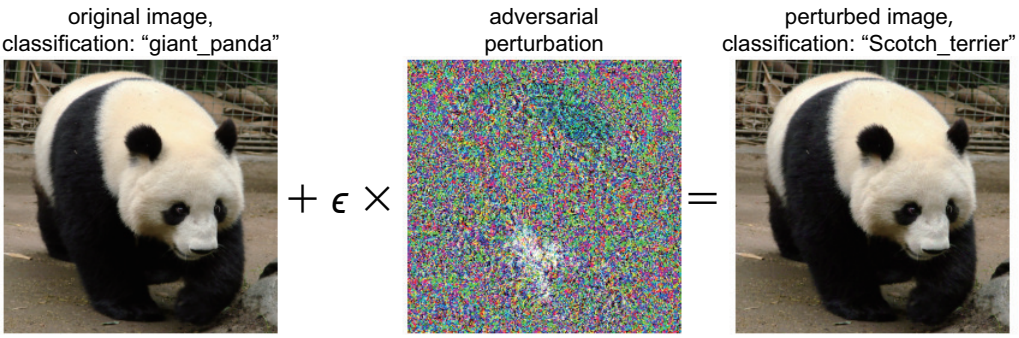

Figure 2. An adversarial perturbation on an image of ImageNet dataset with DNN. This perturbation causes the image to be misclassified by DNN. Left: an input of the model is the original image with its label "giant_panda". Middle: an adversarial perturbation is controlled by ϵ. Right: the perturbed image with the predicted output "Scotch_terrier".

Sign Method (FGSM) [21], Projected Gradient Descent (PGD) [30], C&W attack [31], etc., which easily fool a neural network even when its accuracy is high. These algorithms are designed based on a gradient of the loss function to minimize the perturbation. They thus make adversarial images hard to distinguish from natural images. There are two types of assumptions of the attacker's understanding that are white-box and black-box [32]. In white-box attacks, attackers can access the whole inputs, outputs, architecture, and parameters of a model. Otherwise, attackers only access inputs and outputs and know nothing about the architecture and parameters of a model in black-box attacks [33]. Furthermore, if based on a goal of misclassification, we have two other kinds of attacks, including targeted and non-targeted attacks. Non-targeted attacks mean the adversary wants the classifier to predict any incorrect label without caring what the new classification is. In contrast, the adversary aims to change the output prediction to a specific target class in targeted attacks [34]. The fact that defending a DNN from white-box and non-targeted attacks is more challenging than the defense from black-box and targeted counterparts, but white-box attacks are less to happen in practical systems [35].

The existence of adversarial instances to the image classification task exposes a weakness of DNNs, then how DNNs can learn the robust representation against

adversarial attacks is a big question to the research community of deep learning security. Studying the robustness of DNNs not only gains the security of deep learning models but also helps us explore their missing aspects. Recently, many effective defenses have been proposed, for instance, defensive distillation [36], data augmentation [37], feature denoising networks [38], adversarial detection [39, 40, 41], robust activation functions [5, 42, 43, 44, 45], and adversarial training [21, 30, 31, 46, 47]. Among them, adversarial training has been the most outstanding defense of convolutional neural networks (CNNs), in which CNNs learn the features of adversarial examples in the training procedure. Moreover, [48] proved that the ensemble approach of several trained networks can enhance the robustness, but also increase the model size. From this perception, the random self-ensemble method [49] has been proposed by adding a random noise layer before each convolutional layer to CNNs in both the training and inference phase. Although this algorithm is equivalent to the ensemble of an infinite number of random networks, it is not an optimal way to generate randomness. Hence, the framework of Bayesian Neural Network (BNN) has been learned to improve the robustness of neural networks [50]. With stochastic properties of BNN, it has implied a robust DNN in uncertainty estimation [51, 52, 53, 54, 55, 56, 57, 58]. Liu et al. [50] introduced a combination of adversarial training and Bayesian network to demonstrate that the uncertainty on the weights can be a protection for DNNs. However, they have just executed the experiments on small datasets such as MNIST [59], CIFAR-10 [60]. This method will become more ambitious on large-scale datasets like ImageNet [61, 62].

From the advantages of BNN in the resistance to adversarial attacks, we have proposed a new defense algorithm, named Bayes without Bayesian Learning (BwoBL), which focuses on Bayesian inference. Our method not only achieves robustness against adversarial perturbation of image but also does not consume the additional training cost.

1.2 Research Contribution

Our research contributions can be summarized as follows:

- We assume that the weights of convolutional layers in DNN models are probabilistic distributions to build BNN models. The uncertainty on weights of

BNNs can resist gradient-based attacks and generate the ensemble model to boost the robustness.

- We build BNN models with the parameters that are learned parameters of prominent DNN architectures adjusted by a structural hyperparameter and probabilistic variable.
- We introduce the Bayes without Bayesian Learning algorithm to execute Bayesian inference. BwoBL is shown as a capable defense against various attacks that can be applied for a wide range of DNN architectures, such as CNNs, Self-Attention Networks (SANs), and solve the overhead of the training problem.
- The robustness of our proposal is assessed on CIFAR-10, CIFAR-100, and ImageNet datasets to resist strong attacks, i.e., l_∞ norm PGD and l_2 norm C&W perturbations.
- Our BwoBL algorithm can be easy to integrate into any pre-trained DNN, which is trained on natural and adversarial images to improve the robustness of these pre-trained models without additional training.

These contributions have been published in [63, 64].

1.3 Dissertation Layout

The remainder of this dissertation is structured as follows. Section 2 introduces related work in Bayesian Neural Networks, adversarial attacks, and defense methods. Section 3 describes proposed BNN models, which are based on pre-trained DNNs, and the Bayes without Bayesian Learning algorithm, which focus on Bayesian inference. Section 4 evaluates the robustness of the Bayesian inference approach on deep neural networks and various datasets to resist adversarial attacks. The conclusion is indicated in Section 5.

2 Related Literature

This part introduces the overview of Bayesian Neural Network and its advantages and limitations. Next, adversarial attacks of deep neural networks have been shown. The rest of this part indicates the main lines of defense methods resisting adversarial perturbations.

2.1 Overview of Bayesian Neural Network

Bayesian methods for neural networks have been developed over decades in many different fields. The regularization technique of conventional neural networks can be seen as a Bayesian treatment. The values of regularization coefficients are chosen when using the training data, Bayesian approaches thus avoid the over-fitting problem, which often occurs in the traditional training. For the classification area, DNNs tend to make overconfident decisions, which causes a risk in the prediction of DNNs. For example, DNN-based solutions for diagnostic applications in medicine have widely been developed without any risk management [65, 66]. Meanwhile, reliable information on automated decisions is a key requirement for DNNs. Hence, the integration of Bayesian ideas and DNNs is a principle for estimating the uncertainty of models. Bayesian Neural Network is a robust form of DNNs with a valuable property of uncertainty estimation [67, 68].

BNNs are a unique combination of probabilistic models and neural networks. Thus BNNs can yield stochastic guarantees on their prediction and generate the distribution on their parameters that are learned from the observations [69, 70]. All parameters of BNNs are probabilistic distributions that are single fixed values in DNNs, as shown in Fig. 3. With the application of Bayesian methods for neural networks, we need to find a distribution function $p(\mathbf{w}|D)$ of weights $\mathbf{w}$ while we have observed a dataset D [3, 69, 71]. Bayesian learning for neural networks calculates the posterior distribution of weights, which can be achieved by using Bayes' theorem.

$$p(\mathbf{w}|D) = \frac{p(D|\mathbf{w})p(\mathbf{w})}{p(D)} \tag{1}$$

The denominator $p(D)$ is evidence and given by

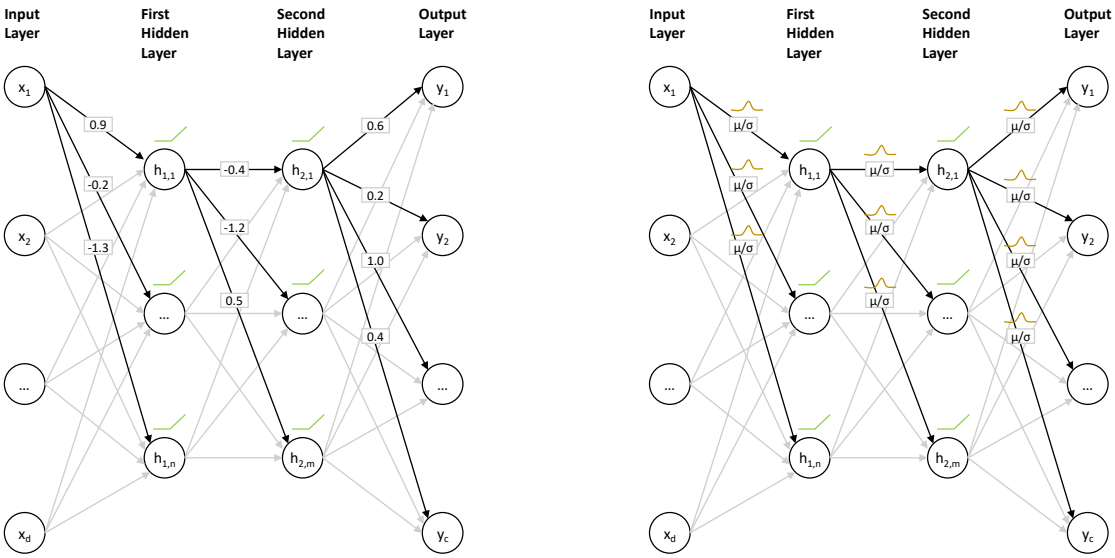

Figure 3. Comparison between a neural network and a Bayesian neural network. Left: the weights of a neural network are single fixed values. Right: the weights of a Bayesian neural network are Gaussian distributions [2].

$$p(D) = \int p(D|\mathbf{w})p(\mathbf{w})d\mathbf{w} \tag{2}$$

The integration of a prior distribution $p(\mathbf{w})$ and a likelihood $p(D|\mathbf{w})$ is exactly intractable. Besides, a large number of parameters makes Bayesian learning hard, the approximation to the posterior distribution is thus used. Various approximation methods for the true posterior distribution have been studied extensively from the '90s. The maximum a posterior scheme and second-order derivatives in the prior distribution to the smoothness of approximating posterior distribution were suggested in [72]. [73] proposed the first variational approximation to the description of weights, which was as a regularization in neural networks. Laplace approximation [74] and the hybrid Monte Carlo method [75] were investigated for training neural networks based on the perturbations of the weights. However, they have been difficult to apply to large size networks in modern applications. Based on the variational inference for neural networks [76], [69] expanded this approximation to show how the gradients can be unbiased and simpler to compute. Many authors have been studied and proposed various schemes to approximate

the intractable posterior probability, for example, dropout [77, 151], Gaussian dropout [78, 79], multiplicative normalization for variational approximation [80], and stochastic gradient MCMC [81]. Currently, Bayes by Backprop [69] is a variational inference method that has successfully been used to train neural networks. In this approach, the posterior distribution is approximated by finding the optimal approximating distribution $q(\mathbf{w}|D)$ that minimizes the Kullback-Leibler (KL) divergence [82] with the true posterior $p(\mathbf{w}|D)$. Variational learning is to find the optimal parameters θ^{opt} of a distribution on the weights.

$$\theta^{opt} = \underset{\theta}{\operatorname{argmin}} \, \mathbf{KL}\left[q_\theta(\mathbf{w}|D) \parallel p(\mathbf{w}|D)\right] \tag{3}$$

where

$$\mathbf{KL}\left[q_\theta(\mathbf{w}|D) \parallel p(\mathbf{w}|D)\right] = \int q_\theta(\mathbf{w}|D) \log \frac{q_\theta(\mathbf{w}|D)}{p(\mathbf{w})p(D|\mathbf{w})} \tag{4}$$

The KL divergence is also intractable to compute exactly, [69] hence approximate the tractable cost function as Eq. (5), which is optimized during training. This approximation is by using Monte Carlo sampling [83, 84] to sample the weights $\mathbf{w}$ from the variational distribution $q_\theta(\mathbf{w}|D)$.

$$F(D, \theta) = \sum_{i=1}^{n} \log q_\theta(w^{(i)}|D) - \log p(w^{(i)}) - \log p(D|w^{(i)}) \tag{5}$$

in which $w^{(i)}$ is the ith Monte Carlo sample drawn from $q_\theta(\mathbf{w}|D)$ and n is the number of samples.

Gaussian variational posterior $\mathcal{N}(\mu, \sigma^2)$ is also supposed on the parameters of BNN in [69], where the weights $\mathbf{w}$ can be sampled by a unit Gauss $\xi \sim \mathcal{N}(0, I)$, shifted by a mean μ and scaled by a standard deviation σ as follows:

$$\mathbf{w} = \mu + \sigma \odot \xi \tag{6}$$

where $\odot$ denotes a point-wise multiplication. From this approximation, we can see that the variational parameter $\theta = (\mu, \sigma)$ is learned during the training phase. It means both the mean and the standard deviation are calculated the gradients by backpropagation, which makes the learning of BNNs bulky. Moreover, we face the generation of representative samples of the weights that is generally not easy in the learning process.

2.2 Adversarial attacks of deep neural networks

Adversarial examples have firstly been indicated by Szegedy et al. [23] in the image classification domain. They proved that a prediction of a state-of-the-art deep neural network could be changed by an imperceptible perturbation to a test image, as seen in Fig. 4. This adversary is found by optimizing the input data to maximize the error function.

We assume that x is an original input with a correct label y. Attack methods are able to seek an adversarial input x' that is classified by a label $y' \neq y$. x and x' are close according to l_p distance metric $\| \cdot \|_p$ noted as:

$$\| x - x' \|_p = \left(\sum_{i=1}^{n} | x_i - x'_i |^p \right)^{\frac{1}{p}} \tag{7}$$

With this attribute, x' is called a non-targeted adversarial example. Instead of classifying x to the prediction $y' \neq y$, we search x' to classify it as a given target class t so that $y' = t$ in targeted adversarial attacks [31]. For attackers, non-targeted attacks are less strong than targeted counterparts. For defenders, by contrast, resistance to non-targeted adversaries is more challenging. Furthermore, Biggio et al. [85] introduced two attack scenarios, i.e., perfect knowledge, and limited knowledge. In a perfect-knowledge setting, the adversary knows the feature space, the output of the classifier, and the trained model, which is also called white-box adversaries. In limited-knowledge adversaries or black-box adversaries, the adversary only knows the input and the output of models. In this dissertation, we focus on the white-box and non-targeted attack setting to baseline models.

Although DNNs have obtained admirable performances in image classification, adversarial attacks can still easily fool the network, even when its accuracy is high. In recent years, several attack approaches have been studied [21, 24, 30, 31, 86, 87, 88, 89] that craft adversarial examples by using a gradient of the objective function in regard to the input data and adjusting to maximize the loss. FGSM [21] is a single-step attack algorithm, which perturbs the original input by the direction of the gradient of the loss function $J(\theta, x, y)$, in which θ is the parameters of a model. A perturbation ϵ is added to each pixel to control the l_∞ distance

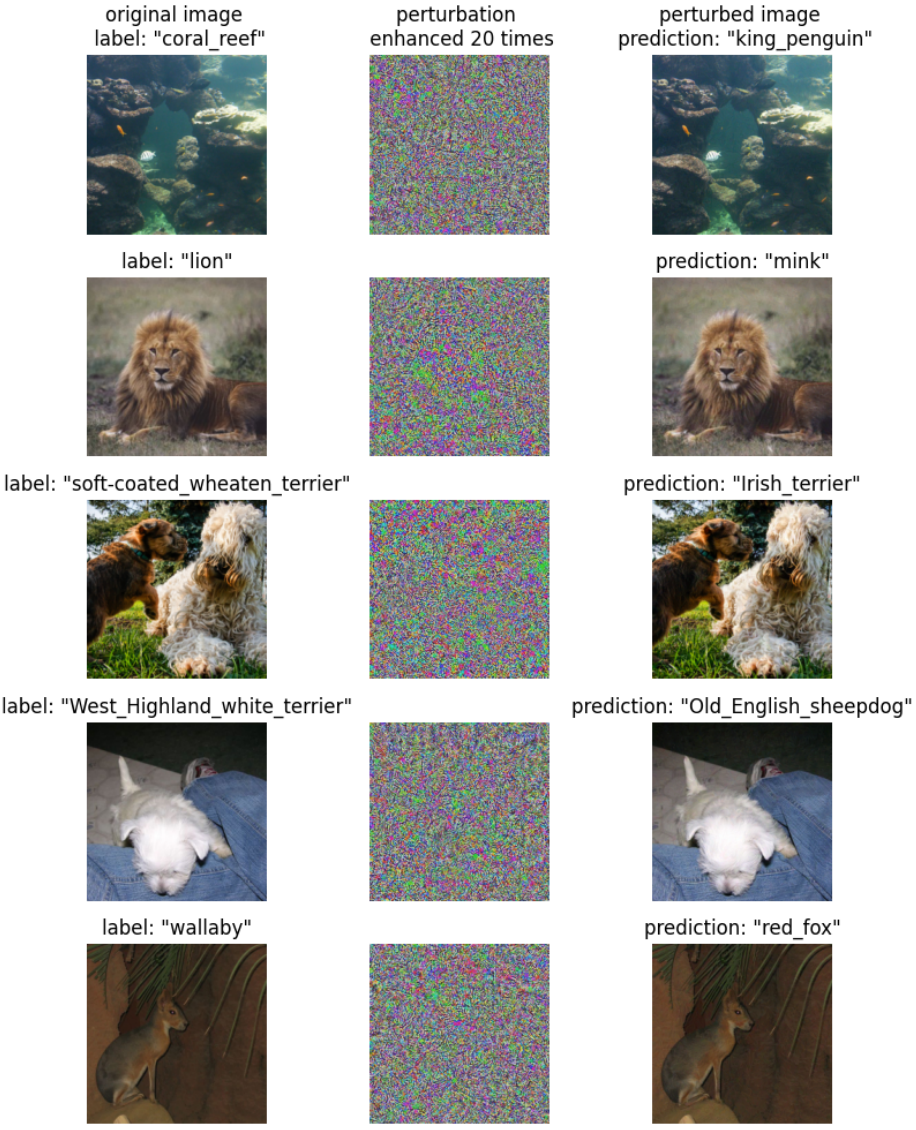

Figure 4. An illustration of adversarial attacks on ImageNet dataset and ResNet-50 architecture. Left: Original images with their labels. Middle: Perturbation enhanced 20 times by PGD attacks. Right: Perturbed images are incorrectly predicted.

metric between the original image x and the perturbed image x' so that

$$\| x - x' \|_\infty \leq \epsilon \tag{8}$$

It is noted that FGSM is designed to be fast rather than optimal. It means this algorithm does not produce minimal adversarial examples. Instead, Kurakin et al. [24] proposed the Iterative Gradient Sign to take a multi-step perturbation based on FGSM. Currently, a PGD attack of Madry et al. [30] is the best multi-step variant of FGSM. The PGD algorithm implements a strong iterative attack to generate adversarial instances, which follows the update as:

$$x^{t+1} = \prod_{\epsilon} \left(x^t + \beta \operatorname{sign} \left(\bigtriangledown_x J(\theta, x, y) \right) \right) \tag{9}$$

in which β is an attack step size, and $\prod_{\epsilon}$ is a projection to l_∞ norm adversary. Other approaches like FGSM have also been demonstrated in [90]. Nonetheless, PGD is still a standard method for large-scale constrained optimization. Madry et al. [30] proved that a network was trained to be robust against PGD attacks, it would become robust to a wide range of first-order adversaries.

In addition, Carlini and Wagner introduced C&W approach [31] that constructs adversarial examples $x' = \frac{1}{2}\left(\tanh(w) + 1\right)$ by searching for w in l_2 distance to solve the optimization problem:

$$\min \left\| \frac{1}{2}\left(\tanh(w) + 1\right) - x \right\|_2^2 + c \cdot f\left(\frac{1}{2}\left(\tanh(w) + 1\right) \right) \tag{10}$$

where c is a constant that is chosen by the modified binary search and $f(\cdot)$ is an objective function defined as:

$$f(x') = \max\left(\max\left(Z(x')_i : i \neq t\right) - Z(x')_t, -\kappa\right) \tag{11}$$

The parameter κ helps us find an adversarial example with high confidence. $Z(\cdot)$ is the output of the network. To select an optimal c, we must perform several iterations of binary search. With each chosen value of c, we also run a number of iterations of gradient descent to seek an optimal adversarial instance. Owing to these iterations, the converge of C&W attacks is slower than that of PGD. Therefore, in the experiments, we execute PGD as the main attack and C&W is the additional evaluation to verify our algorithm.

2.3 Defense methods against adversarial perturbation

As long as DNNs can be vulnerable to adversarial attacks, a large number of defense algorithms have been proposed in recent times [30, 31, 46, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100]. Several authors showed the properties of activation functions could affect the robustness of DNNs and designed robust activation functions as defense methods. However, most of them are optimized by adversarial training. It is known that adversarial training is currently the most robust algorithm that trains DNNs on perturbed examples. Madry et al. [30] proposed adversarial training to improve the robustness of DNNs resisting PGD attacks on MNIST and CIFAR-10. They focus on the natural saddle point (min-max) formulation that is optimized by adversarial training as follows:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim D} \left[\max_{\epsilon \in S} J(\theta, x + \epsilon, y) \right] \tag{12}$$

where D is an underlying data distribution over the pairs of x examples and y corresponding labels, $J(\cdot)$ is a suitable loss function, θ is network parameters, and the per-pixel perturbation ϵ is allowed in the perturbed range S.

The formula in Eq. (12) shows two computation steps of adversarial training: (1) an inner maximization, which takes adversarial instances, and (2) an outer minimization, which finds model parameters. The robustness of adversarial training depends on the strength of the adversaries that are generated by the inner maximization. There are two problems in the generation of adversarial examples for training, which are derived from Eq. (12).

- Firstly, if ϵ is fixed during training, the robustness of the network only resists an adversarial attack of that ϵ value.
- Secondly, many iterations of the gradient computation are required so that PGD algorithm converge to optimal adversaries, which consumes the high cost of the computation time.

Due to the high cost of generating adversarial instances, many FGSM-based adversarial training methods are designed to accelerate the computation against PGD attacks. Nevertheless, the fast training on FGSM adversaries is only robust against non-iterative attacks but is difficult to resist iterative counterparts like

PGD adversaries. Besides, Wong et al. [98] have shown the lack of a diversity of adversarial examples generated by FGSM attack causes catastrophic overfitting in the training phase. Additionally, to achieve the outer minimization in Eq. (12), the training process must be performed in many epochs. The iteration of the inner maximization and the outer minimization make adversarial training unrealistic on large-scale datasets like ImageNet.

The idea of adding stochastic components to DNNs and training these models on perturbed data has been perceived as a good defense to prevent strong gradient-based attacks. Liu et al. [50] indicated the dominance of BNNs in improving the robustness of DNNs to adversarial attacks on CIFAR-10 and ImageNet-143. However, adversarial Bayesian training will be harder on ImageNet because Bayesian learning has some disadvantages as below,

- The weights of BNNs are described by probabilistic distributions, e.g., Gaussian function. The number of parameters is thus double. If a network architecture is sizable, it shall have a huge number of parameters.

- Since the functional form of BNNs does not enable it to exact the integral, we often take a variational approximation to the posterior distribution on the weights. This makes the optimization problem much larger scale.

- The uncertainty of the weights leads to a coherent variability of the training data during training. BNN learning hence trains an ensemble of the networks instead of a single network, which is called Ensemble Learning. This is not tractable for BNNs of any practical size.

The current trend of defense approaches must be both the robustness improvement and the overhead limitation of the training problem. Based on the advantages of BNNs towards resisting perturbations, we mainly focus on Bayesian inference with the proposed BwoBL algorithm to withstand strong adversarial attacks and avoid the additional adversarial training.

3 Bayesian Neural Networks and Bayesian inference

Deep convolutional neural networks have currently been the state-of-the-art models for image classification [101, 102, 103, 104, 105, 106, 107, 108, 109, 110]. Additionally, recent work has shown self-attention as a capable alternative for image recognition models [111, 112, 113, 114, 115, 116, 117, 118, 119, 120]. Based on these pre-trained CNNs and SANs, we build our BNNs that replace convolutional layers in CNNs and linear transformation layers in SANs with Bayes layers. In Bayes layers, instead of single-point values, the parameters are the probabilistic distribution. Other layers in CNN and SAN architectures are originally kept.

3.1 Proposed BNNs

There are many definitions of Bayesian methods for neural networks. For example, the regularization technique of conventional neural networks or adding dropout layers to neural networks can be seen as a Bayesian treatment [77, 151]. Neural networks with the probabilistic distribution over the weights have been studied as Bayesian neural networks [52, 57], which are described in Fig. 5. When we generate the uncertainty on the model parameters, our model is a Bayesian neural network. Several distribution functions, which are used to describe the parameters of a Bayesian neural network, are Gauss, Bernoulli, etc. The Gaussian distribution is widely utilized in Bayesian neural networks due to its more accurate prediction, although the number of parameters and the calculation increase.

We have calculated the posterior distribution of weights according to Bayes' theorem in the Bayesian learning process, as introduced in Section 2.1. However, Bayesian probability theory often comes with an intractable computation because the number of parameters is very large and the function form does not allow it to exact integration. Then, various approximations have been employed, in which variational inference is a new approximation for Bayesian neural networks. In this Bayesian learning, the variational posterior of weight is supposed to be a Gaussian distribution, then, the variational posterior parameters are μ and σ. To approximately calculate, Monte Carlo sampling is utilized to draw the sample of

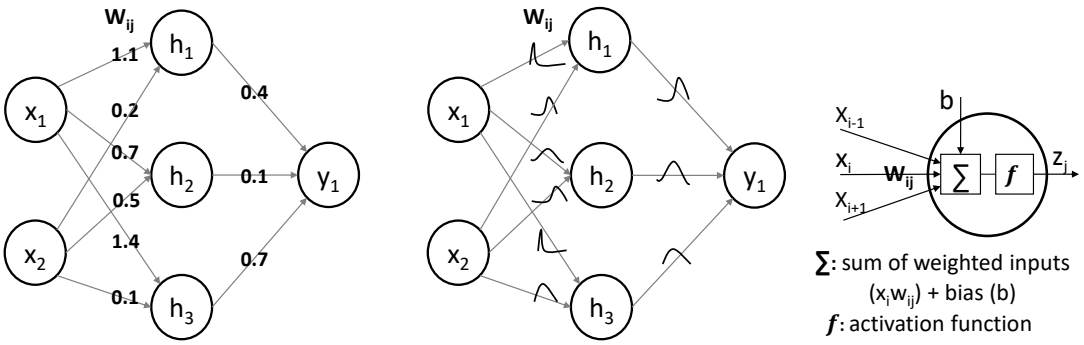

Figure 5. A comparison between a DNN and a BNN and an inner structure of a neuron that show the replacement of fixed values with the probabilities on the weights w_{ij}. *Left*: a DNN with specific values of the weights. *Middle*: a BNN with the distribution function of the weights. *Right*: an inner structure of a neuron with the weights w_{ij} that are fixed values in a DNN replaced by probabilistic distributions in a BNN.

weights from the Gaussian variational posterior, as shown in Eq. (6). Therefore, we have just initialized the samples of weights according to Eq. (6) to build our Bayesian model.

The Bayes layer is the main component in our BNN models, which is described by the probabilistic distribution on parameters. In pre-trained CNNs, we substitute Bayes-conv layers for convolutional layers with the same size of filters. Linear transformation layers are replaced by Bayes-linear layers in pre-trained SANs. As an application of Gaussian variational posterior [69] and Gaussian dropout [78], we approximate the variational posterior of our BNNs with a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$. Based on the relationship of parameters between DNNs and BNNs (as seen in Fig. 5), we employ pre-trained DNNs to construct our BNNs in which the mean μ of weights is assigned by single-point weights of DNNs. We focus on controlling the variance σ^2 to create uncertainty on the weights.

We assume the posterior approximation on the weights to be fully factorized Gaussian distribution $q(\mathbf{w}) = \mathcal{N}(\theta,\, \rho\theta^2)$ that has been discussed by Kingma et al. [79]. This variational posterior is accurately an approximation to implement a reparameterization in Bayes layers, which plays an essential role in the construction of our BNNs. The variance of $\mathbf{w}$ is tied by its magnitude. A larger

weight is then valuable when it is robust to noise. From this perspective, we have a formula of the weight sample in Bayes-conv and Bayes-linear layers instead of Eq. (6) as follows

$$\mathbf{w} = \theta + \alpha\theta \odot \xi \tag{13}$$

where ξ is a Gaussian unit $\mathcal{N}(0, I)$. We treat θ as single fixed values of pre-trained DNN parameters while $\alpha = \sqrt{\rho}$ is empirically determined by the robustness of the BNN model against the adversarial attack. Our aim is the use of pre-trained DNNs to construct our BNNs and avoid Bayesian learning, which is a term of transfer learning. From Fig. 5(right), we have the relationship between input x_i, output z_j and parameters as follows:

- In DNNs: $z_j = f\left(\sum_0^{m-1} \theta_{ij} x_i + b\right)$
- In BNNs: $z_j = f\left(\sum_0^{m-1} \mathrm{w}_{ij} x_i + b\right)$

where $b \in \mathbb{R}$ is bias, θ_{ij} is single-point values of the weights, the output z_j is also single fixed values in DNNs. Contrarily, in BNNs, the weights w_{ij} is the random samples of Gaussian distribution, the output z_j is then an uncertainty followed w_{ij}. Therefore, the output of BNN is stochastic values, which is generated by the uncertainty on weights w_{ij} and make the ensemble inference procedure of BNN.

It is known that DNNs work well on the training and test data, which are in the same feature space and distribution. When these features and distributions change, DNNs must be rebuilt from the new training data. It is too expensive for the reconstruction of DNNs in many real-world applications. Accordingly, transfer learning has been verified to make use of the well-learned knowledge from source datasets [121, 122, 123, 124, 125]. The requirement for transfer learning may arise when the datasets are getting bigger and bigger. In image recognition, transfer learning has been widely used in real datasets to boost the performance of DNNs [126, 127, 128, 129, 130]. For this scenario, fine-tuning the pre-trained networks on large-scale datasets, such as ImageNet, is taken up. If original ImageNet is the source domain, adversarial ImageNet can be seen as the target domain. Hence, we can apply transfer learning to reuse learned parameters of DNNs from the source domain to the target domain and reduce the limitations of training.

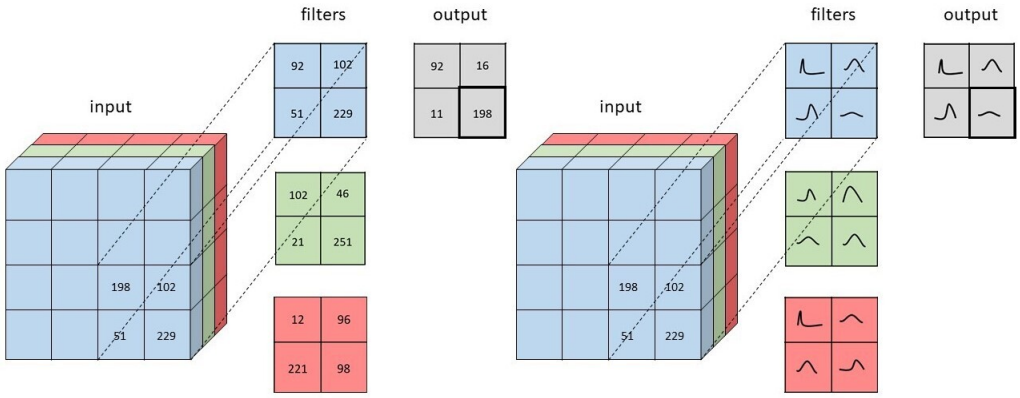

Figure 6. An illustration of convolutional filters in CNN and BCNN. Left: weights of CNN are single-point estimates. Right: weights of BCNN are probabilistic distributions [3].

From Bayesian methods for neural networks, we build Bayes-conv and Bayes-linear layers followed by Eq. (13), in which θ is learned parameters of pre-trained DNNs. We mainly concentrate on adjusting α that tunes the standard deviation σ to create uncertainty on the parameters of BNNs. After constructing BNNs, we conduct Bayesian inference via BwoBL algorithm towards robustness to adversarial examples.

3.2 Construction of BNN based on CNN

Building Bayesian convolutional neural networks (BCNNs) with the probabilistic distribution on weights of convolutional layers has been conducted by Shridhar et al. [3], as shown in Fig. 6.

Whereas the frequentist inference of CNNs that has only one convolutional operation on the filters with the single-point estimate, BCNNs are applied two convolutional operations to weights determined by a Gaussian distribution, i.e., the mean μ and the variance σ^2. Therefore, the first convolution is the mean μ of the variational distribution, which is treated as a single fixed estimate in the frequentist inference. The second is the standard deviation σ, which controls the weight uncertainty. Because the mean is fixed values of pre-trained CNNs, the second convolution plays an important role in the performance of BCNNs.

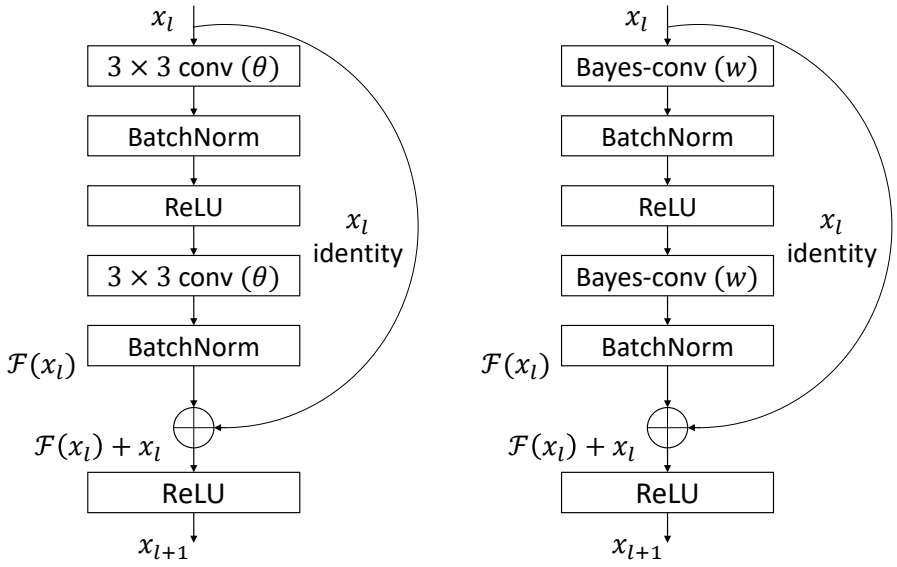

Figure 7. Convolutional layers (3×3) in a building block of ResNet model are replaced with Bayes-conv layers (3×3) in our BNN networks. We keep the other layers of ResNet, such as batch normalization, activation function (ReLU). Left: a building block of ResNet architecture. Right: a building block of our Bayesian ResNet.

We realize that the weight transformation of convolutional layers is good enough for generating Bayesian models but it does not enlarge the model parameters. All convolutional layers of pre-trained CNNs are replaced by Bayes-conv layers in our BNNs with the sample of parameters in Eq. (13). In our experiments, we utilize the outstanding DNNs that achieve the advanced performance on ImageNet currently, i.e, ResNet [131], PreActResNet [132], and EfficientNet [4] to construct the robust BNNs.

3.2.1 Bayesian ResNet

Residual Network (ResNet) won 1st place in the ILSVRC 2015 classification competition on ImageNet. He et al. [131] addressed a deep residual learning framework to few stacked layers and made a building block defined as

$$y = \mathcal{F}\left(x, \{W_i\}\right) + x \tag{14}$$

where x and y present the input and output of stacked layers. $\mathcal{F}(x, \{W_i\})$ is the residual mapping. A building block of two convolutional layers is illustrated in Fig. 7 (left), which is the basic blocks in ResNet-18 (18 layers), ResNet-20 (20 layers), and ResNet-34 (34 layers). $\mathcal{F}(x) = W_2\sigma(W_1x)$ in which σ is a non-linearity ReLU [133]. $y = \mathcal{F}(x) + x$ is operated by a shortcut connection, which is added to each pair of 3×3 filters, and element-wise addition. The second ReLU $\sigma(y)$ is adopted after this addition.

He et al. [131] also described deeper bottleneck architectures by stacking bottleneck building blocks, as shown in Fig. 8 (left). Identity shortcuts that are designed for bottleneck blocks bring more efficient models, for example, ResNet-50 (50 layers), ResNet-101 (101 layers), ResNet-152 (152 layers). Additionally, Xie et al. [134] proposed ResNeXt models, which tend to the width of ResNets. For example, we use ResNeXt-101-32×8d that is 2× deeper and 4× wider than ResNet-50. These are state-of-the-art models with very competitive accuracy on ImageNet recently.

Based on ResNet architectures, we construct our BNN by replacing all convolutional layers with Bayes-conv layers, which is demonstrated in Figures 7 (right), 8 (right), and 9 (right). In Figures 7 and 8, we visually illustrate the construction of Bayesian non-bottleneck and bottleneck blocks. After forming each building block, we assemble the whole model. For instance, our Bayesian ResNet-18 is made from the original ResNet-18, which is shown in Fig. 9.

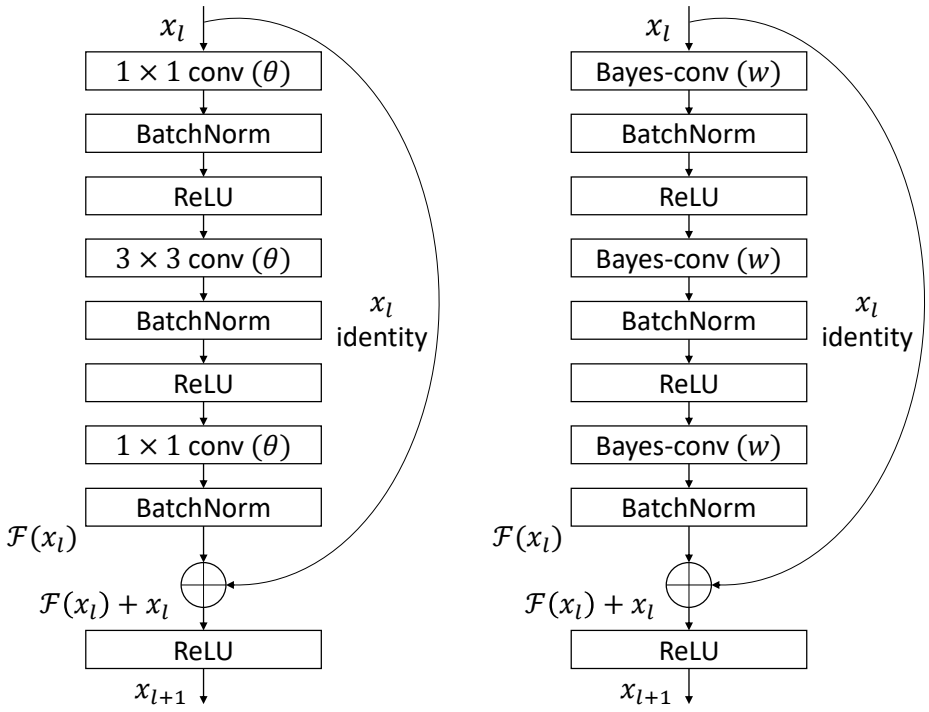

Figure 8. Convolutional layers in a "bottleneck" block of ResNet model are replaced with Bayes-conv layers in our BNN networks with the same dimension. Left: a "bottleneck" building block of ResNet-50/101/152. Right: a "bottleneck" building block of our Bayesian ResNets.

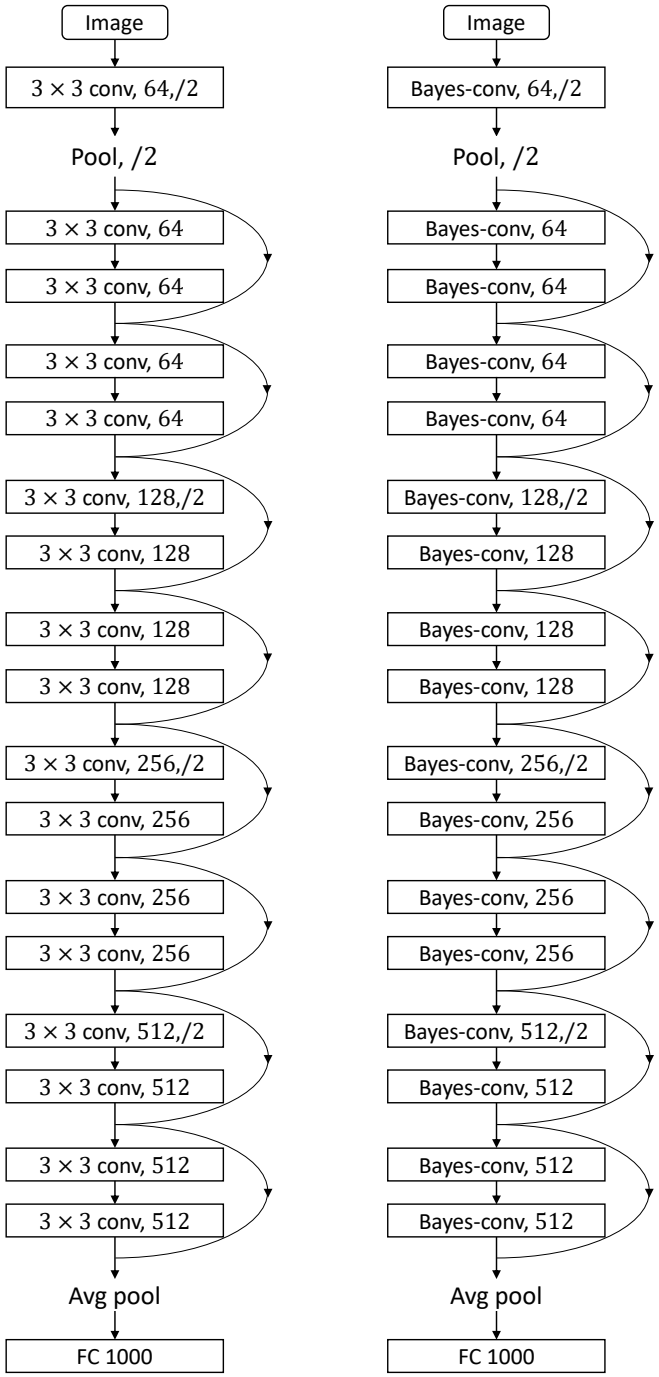

Figure 9. Bayesian ResNet-18 is built on the original ResNet-18. Bayes-conv layers of Bayesian ResNet-18 (right) are the alternative of convolutional layers in the original ResNet-18 (left). Other layers (batch normalization and ReLU) of original model are kept and hidden.

3.2.2 Bayesian PreActResNet

In PreActResNet architectures, He et al. [132] deeply analyzed the role of identity mapping in deep residual learning of ResNet towards the view of activation functions, including ReLU and Batch Normalization (BN) [135]. They indicated that re-arranging the activation functions (ReLU and BN) could get competitive results on CIFAR-10/100.

In ResNet, BN is arranged after each weight layer, and ReLU is used after BN, excepting that the last ReLU in a building block is after element-wise addition. It is thus called "post-activation", as seen in Figures 7 and 8. Particularly, let x_l and x_{l+1} be the input and output of the l-th unit, Eq. (14) is rewritten as

$$\begin{aligned} y_l &= \mathcal{F}\left(x_l, W_l\right) + x_l \\ x_{l+1} &= f\left(y_l\right) \end{aligned}$$

where f is a ReLU function. Accordingly, the activation affects the entire shortcut connection, especially in backward propagation. So as to reduce this influence of the activation function, a "pre-activation" design has been proposed in [132] with the following formula:

$$x_{l+1} = \mathcal{F}\left(f\left(x_l\right), W_l\right) + x_l \tag{15}$$

Therefore, both BN and ReLU are used as pre-activation in PreActResNet models, which is illustrated in Fig. 10 (left). This design is slightly better than ResNet on CIFAR-10/100 as pointed out in [132]. We also apply Bayes-conv layers to pre-trained PreActResNet to produce Bayesian PreActResNet (Fig. 10 (right)) that is against the adversarial attack on CIFAR-10 in our experiments.

3.2.3 Bayesian EfficientNet

EfficientNet is a family of deep convolutional neural networks with a scaling method on all dimensions of depth/width/resolution [4]. These networks are based on scaling up MobileNets [110, 136] and ResNets [131].

Tan et al. [4] proposed an effective compound coefficient, which balanced the depth, width, and resolution of networks to improve the performance of DNNs, as seen in Fig. 11. They indicated that deeper networks could capture more complicated features [108, 109, 131]. Wider networks can capture more fine-grained

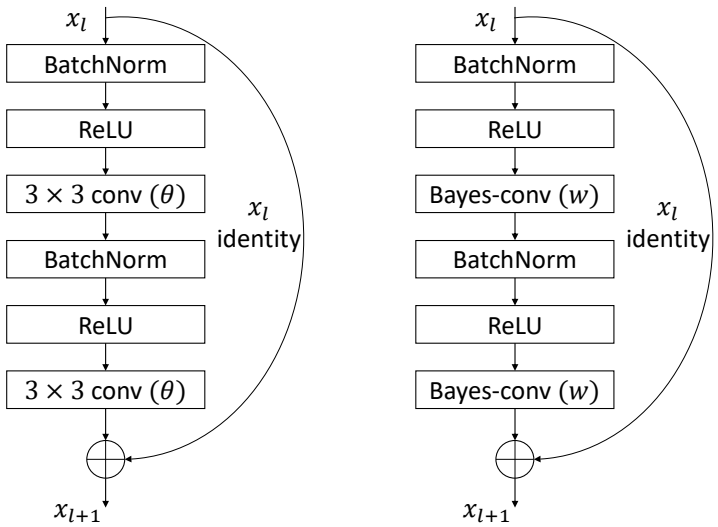

Figure 10. A "pre-activation" block (left) in PreActResNet architecture is altered by a Bayesian "pre-activation" block (right) in our BNN model. The replacement is only in convolutional layers with keeping the size of filters.

features and are easily trained [136, 137, 138]. DNNs with higher resolution of input images can capture more fine-grained features to achieve better accuracy, for instance, Inception with 299×299 resolution [109], NASNet with 331×331 resolution [139], Gpipe with 480×480 resolution [140] for ImageNet, 600×600 resolution for object detection [141, 142].

The baseline network of [4] is EfficientNet-B0, which is inspired by MnasNet [137]. Table 1 presents the architecture of EfficientNet-B0, with its building block is that mobile inverted bottleneck "MBConv". Each MBConv block is constituted from sub-blocks. Each sub-block is a combination of modules (in Fig. 12). The detailed architecture of EfficientNet-B0 is explained in Fig. 13. Scaling up the baseline network with different compound coefficients to have EfficientNet-B1 to B7 that comprise the modules in Fig. 12.

Our Bayesian EfficientNets are formed by the replacement of convolutional, depthwise convolutional, and pointwise convolutional layers in basic modules of EfficientNets with our Bayes-conv layers. EfficientNets are a large structure with hundreds of layers, it thus is not easy to train. The use of pre-trained EfficientNets to build Bayesian EfficientNets promises better performance in realistic datasets.

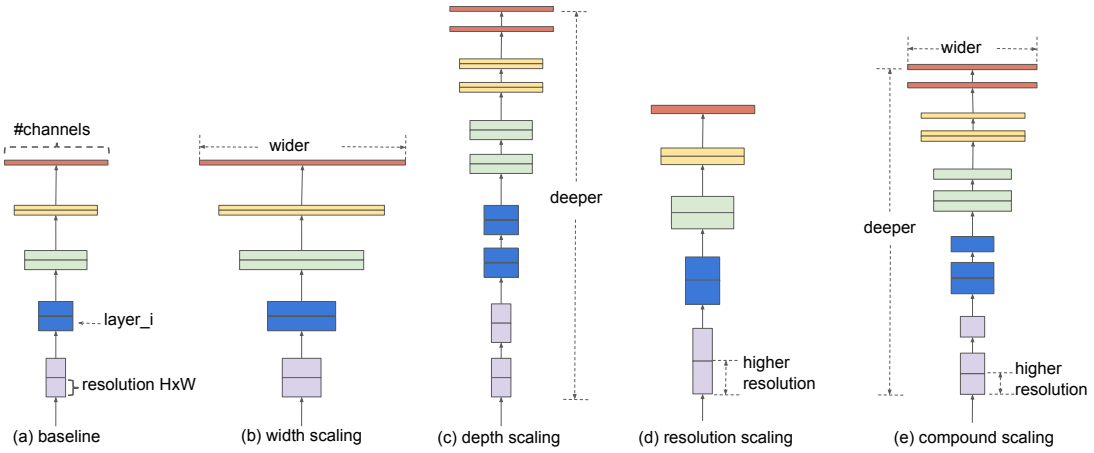

Figure 11. Network scaling method [4]. (a) a baseline network; (b), (c), and (d) are scaling on the width, depth, and resolution of a baseline network; (e) a compound scaling method of EfficientNets.

Table 1. Architecture of EfficientNet-B0 and the corresponding substitution between EfficientNet and Bayesian EfficientNet.

Stage i	**Operator** $\mathcal{F}_i$	**Resolution** $H_i \times W_i$	**Channel** C_i	**Layers** L_i
1	Conv 3×3	224×224	32	1
2	MBConv1, k3×3	112×112	16	1
3	MBConv6, k3×3	112×112	24	2
4	MBConv6, k5×5	56×56	40	2
5	MBConv6, k3×3	28×28	80	3
6	MBConv6, k5×5	14×14	112	3
7	MBConv6, k5×5	14×14	192	4
8	MBConv6, k3×3	7×7	320	1
9	Conv 1×1 & Pooling & FC	7×7	1280	1

EfficientNet	**Bayesian EfficientNet**
Conv 3×3	Bayes-conv 3×3
MBConv1, k3×3	Bayes-MBConv1, k3×3
MBConv6, k3×3	Bayes-MBConv6, k3×3
MBConv6, k5×5	Bayes-MBConv6, k5×5

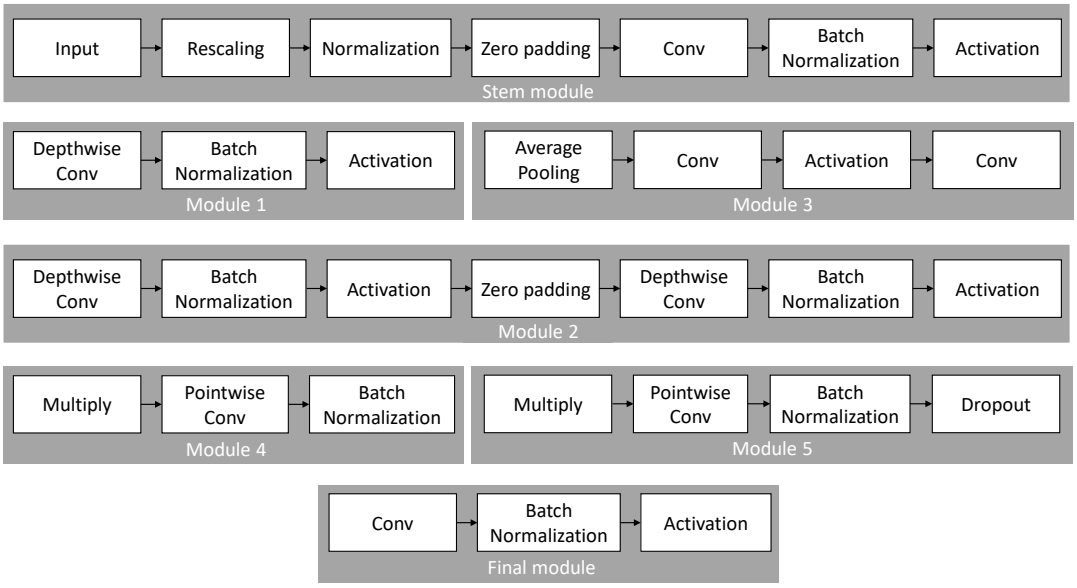

Figure 12. Basic modules of EfficientNets. Bayesian EfficientNets are built by the replacement of *conv, depthwise conv, poinwise conv layers* with *Bayes-conv layers*. Other layers are originally kept.

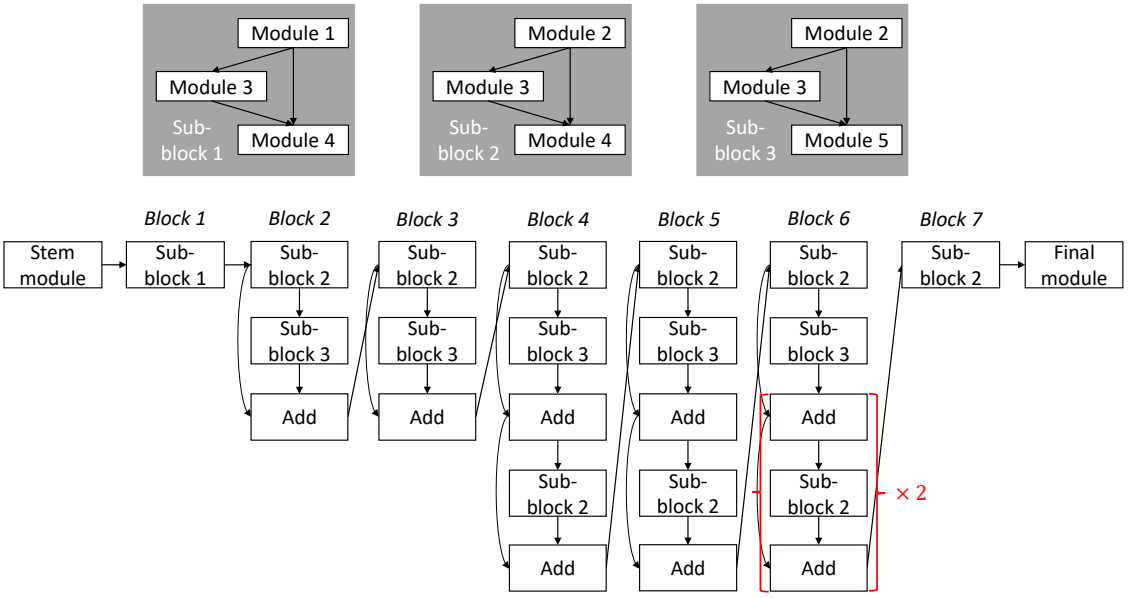

Figure 13. Detailed architecture of EfficientNet-B0.

3.3 Construction of BNN based on SAN

Recent work has shown effective architectures [111, 113, 115, 119, 120] that employ a self-attention mechanism as an alternative for convolutional networks in image classification.

Zhao et al. [115] explored two types of self-attention networks. One is pairwise, the other is patchwise. In particular, the pairwise self-attention derives the standard dot-product attention, as follows:

$$y_i = \sum_{j \in \mathcal{R}(i)} \alpha\left(x_i, x_j\right) \odot \beta\left(x_j\right) \tag{16}$$

where i is the location of vector x_i in the feature map. The footprint $\mathcal{R}(i)$ is a set of indices to specify the feature vector, which is aggregated to the new feature y_i. The function α computes the weights $\alpha\left(x_i, x_j\right)$ that are combined with the transformed features $\beta\left(x_j\right)$ and are decomposed as below:

$$\alpha\left(x_i, x_j\right) = \gamma\left(\delta\left(x_i, x_j\right)\right) \tag{17}$$

The relation function δ makes a single vector of the features x_i, x_j. The function γ maps this vector to a vector that is connected to $\beta\left(x_j\right)$. The function δ is explored by the following forms, in which φ and ψ are trainable transformations, e.g., linear mappings.

- Summation: $\delta\left(x_i, x_j\right) = \varphi(x_i) + \psi(x_j)$
- Subtraction: $\delta\left(x_i, x_j\right) = \varphi(x_i) - \psi(x_j)$
- Concatenation: $\delta\left(x_i, x_j\right) = [\varphi(x_i), \psi(x_j)]$
- Hadamard product: $\delta\left(x_i, x_j\right) = \varphi(x_i) \odot \psi(x_j)$
- Dot product: $\delta\left(x_i, x_j\right) = \varphi(x_i)^\intercal \psi(x_j)$

The patchwise self-attention is a class of operators, which can identify specific locations within their footprint. The patchwise has the following form:

$$y_i = \sum_{j \in \mathcal{R}(i)} \alpha\left(x_{\mathcal{R}(i)}\right)_j \odot \beta\left(x_j\right) \tag{18}$$

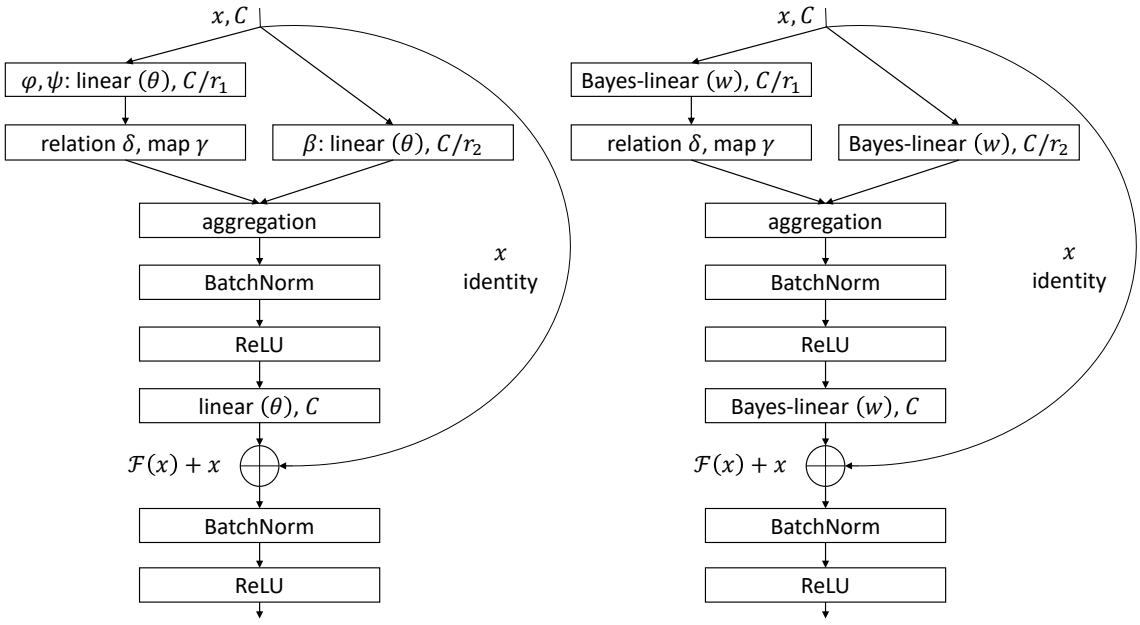

Figure 14. Linear transformation layers of the self-attention block (left) in SAN models are replaced by Bayes-linear layers of our BNNs (right). Other layers of SAN architecture are originally maintained.

where $x_{\mathcal{R}(i)}$ is a patch of feature vectors in the footprint $\mathcal{R}(i)$. $\alpha\left(x_{\mathcal{R}(i)}\right)_j$ is a vector at location j in the tensor $\alpha\left(x_{\mathcal{R}(i)}\right)$, which is spatially corresponding to the vector x_j. The weight computation $\alpha\left(x_{\mathcal{R}(i)}\right)$ can be decomposed as follows:

$$\alpha\left(x_i, x_j\right) = \gamma\left(\delta\left(x_{\mathcal{R}(i)}\right)\right) \tag{19}$$

The function γ maps a vector that is yielded by $\delta\left(x_{\mathcal{R}(i)}\right)$ to a tensor. This tensor consists of weight vectors for all locations j. The function δ combines the vectors x_j from the patch $x_{\mathcal{R}(i)}$, and can be explored as below:

- Star-product: $\delta\left(x_{\mathcal{R}(i)}\right) = \left[\varphi(x_i)^\intercal \psi(x_j)\right]_{\forall j \in \mathcal{R}(i)}$
- Clique-product: $\delta\left(x_{\mathcal{R}(i)}\right) = \left[\varphi(x_j)^\intercal \psi(x_k)\right]_{\forall j,k \in \mathcal{R}(i)}$
- Concatenation: $\delta\left(x_{\mathcal{R}(i)}\right) = \left[\varphi(x_i), \left[\psi(x_j)\right]_{\forall j \in \mathcal{R}(i)}\right]$

The self-attention operations can be visually represented in Fig. 14 (left), which is based on the residual blocks of ResNet architecture. Zhao et al. [115]

designed SAN19 architecture (19 self-attention blocks), which corresponded to ResNet-50. We build the Bayesian SAN model by simply replacing linear transformation layers of SAN19 with our Bayes-linear layers. The parameters of Bayes-linear layers are sampled as Eq.(13) with utilizing learned parameters of pre-trained SAN19. The Bayesian self-attention block is shown in Fig. 14 (right). The combination of Bayesian self-attention blocks generates Bayesian SAN19, as indicated in Table 2.

Table 2. SAN19 and Bayesian SAN19 for ImageNet. SA Block, C-d sa, C-d linear stand for a self-attention block, a self-attention operation with the output dimension C, a linear layer with the output dimension C. Linear layers and sa blocks of SANs are substituted by Bayes-linear layers and Bayes-sa blocks.

Layer name	**Output size**	**SAN19**	**Bayesian SAN19**
Input	224×224×3	64-d linear	64-d Bayes-linear
Transition	112×112×64	2×2, stride 2 max pool → 64-d linear	2×2, stride 2 max pool → 64-d Bayes-linear
SA Block	112×112×64	$\begin{bmatrix} 3\times 3, 16\text{-d sa} \\ 64\text{-d linear} \end{bmatrix} \times 3$	$\begin{bmatrix} 3\times 3, 16\text{-d Bayes-sa} \\ 64\text{-d Bayes-linear} \end{bmatrix} \times 3$
Transition	56×56×256	2×2, stride 2 max pool → 256-d linear	2×2, stride 2 max pool → 256-d Bayes-linear
SA Block	56×56×256	$\begin{bmatrix} 7\times 7, 64\text{-d sa} \\ 256\text{-d linear} \end{bmatrix} \times 3$	$\begin{bmatrix} 7\times 7, 64\text{-d Bayes-sa} \\ 256\text{-d Bayes-linear} \end{bmatrix} \times 3$
Transition	28×28×512	2×2, stride 2 max pool → 512-d linear	2×2, stride 2 max pool → 512-d Bayes-linear
SA Block	28×28×512	$\begin{bmatrix} 7\times 7, 128\text{-d sa} \\ 512\text{-d linear} \end{bmatrix} \times 4$	$\begin{bmatrix} 7\times 7, 128\text{-d Bayes-sa} \\ 512\text{-d Bayes-linear} \end{bmatrix} \times 4$
Transition	14×14×1024	2×2, stride 2 max pool → 1024-d linear	2×2, stride 2 max pool → 1024-d Bayes-linear
SA Block	14×14×1024	$\begin{bmatrix} 7\times 7, 256\text{-d sa} \\ 1024\text{-d linear} \end{bmatrix} \times 6$	$\begin{bmatrix} 7\times 7, 256\text{-d Bayes-sa} \\ 1024\text{-d Bayes-linear} \end{bmatrix} \times 6$
Transition	7×7×2048	2×2, stride 2 max pool → 2048-d linear	2×2, stride 2 max pool → 2048-d Bayes-linear
SA Block	7×7×2048	$\begin{bmatrix} 7\times 7, 512\text{-d sa} \\ 2048\text{-d linear} \end{bmatrix} \times 3$	$\begin{bmatrix} 7\times 7, 512\text{-d Bayes-sa} \\ 2048\text{-d Bayes-linear} \end{bmatrix} \times 3$
Classification	1×1×1000	average pool → 1000-d fc → softmax	

3.4 Bayesian inference via BwoBL algorithm

3.4.1 Opportunities for Bayesian inference

In recent years, much research has proved that the ensemble model outperforms the single model, especially in adversarial attacks [49, 50, 99]. It is shown that ensemble inference is a crux of BNNs [55]. The uncertainty of the weights makes BNNs equivalent to the ensemble of random models but do not increase the number of models.

Liu et al. [50] performed an adversarial Bayesian training, which achieves the good performance of BNNs under strong PGD attacks on CIFAR-10. Nonetheless, the challenges of adversarial Bayesian learning make the training of BNNs difficult in real-world applications. To address this issue, we build a BNN model based on a pre-trained DNN, which has already been the state-of-the-art performance in image classification. We apply transfer learning from DNNs to BNNs. The learned parameters of original DNNs are employed to execute Bayesian inference efficiently, but our method does not add any training phase.

We represent a preliminary evaluation to compare the robustness of defensive methods on CIFAR-10 under l_∞ PGD and l_2 C&W attacks in Tables 3 and 4. CIFAR-10 is a small dataset that is easy to implement in most defensive approaches. Table 3 verifies the construction of our BNN from pre-trained CNN and executing Bayesian inference can boost the robustness of the pre-trained model against adversarial attacks without the iterative adversarial training. Besides, our BNN is more robust than BNN trained on adversarial images (Adv-BNN). Table 4 demonstrates the robustness of our algorithm to adversarial attacks compared with robust activation functions when both methods do not perform adversarial training. [5] trained ResNet-20 with various activation functions on natural CIFAR-10 and proved the robustness of their SPLASH activation function. We primarily alter convolutional layers of pre-trained ResNet-20 without changing the activation function and sharply improve the robustness to adversarial attacks. With these preliminary comparisons, the potential of our idea is revealed on the small dataset. The detailed evaluation of a realistic dataset will be implemented in our experiments.

With the rapid deployment of deep learning, pre-trained DNNs are universally

Table 3. Evaluation of top-1 accuracy to compare the robustness of adversarial training by CNN (FAST), adversarial learning by BCNN (Adv-BNN), and our proposal on CIFAR-10 with 10000 images of the validation set, under l_∞ PGD attack with pixel perturbation ($\epsilon = 8/255$) and 20 iterations.

Method	**Network**	**Top-1 Accuracy (%)**
FAST [98]	PreActResNet-18 (CNN)	46.77
Adv-BNN [50]	VGG-16 (BCNN)	47.58
Our proposal	pre-trained PreActResNet-18 + proposal (BCNN)	50.02

utilized and importantly contribute to the research community. We take advantage of pre-trained models to reinforce their resistance to adversarial attacks and avoid the overhead of the training phase. It should be emphasized that leveraging learned parameters of existing models is key in our algorithm.

Owing to no Bayesian learning in our algorithm, we can degrade the complexity of the training phase. Bayesian learning is explained by the Bayes by Backprop algorithm [69] as follows:

1. The variational posterior is supposed to be a diagonal Gaussian distribution. The weights of Bayesian model can be sampled by a unit Gaussian distribution $\xi \sim \mathcal{N}(0, I)$.

2. The posterior sample of weights: $\mathbf{w} = \mu + \sigma \odot \xi$.

3. The ensemble of several forward passes to achieve the output and calculate the loss $f(\mathbf{w}, (\mu, \sigma))$.

4. Calculate the gradient to the mean: $\triangle_\mu$.

5. Calculate the gradient to the standard deviation: $\triangle_\sigma$.

6. Update the variational parameters:

$$\mu \leftarrow \mu - \alpha \triangle_\mu$$
$$\sigma \leftarrow \sigma - \alpha \triangle_\sigma$$

With Bayesian inference, we just perform steps 1, 2, and 3. Steps 1 and 2 are to build the uncertainty on the model parameters. In step 3, we execute several forward passes and achieve the ensemble output. Based on pre-trained models, we copy their learned parameters (θ) and assign them to the mean of weights in

Table 4. A comparison of robust activation functions and our proposal on CIFAR-10 under l_2 C&W attack. We compare the results of SPLASH activation function [5] and our BNN model on 1000 images chosen from correctly classified images of ResNet-20. We also execute our experiment five times to calculate *mean* $\pm$ *standard deviation* of the number of success attacks as [5].

Network	**Activation**	**# of success attacks**
ResNet-20 [5]	ReLU	903 $\pm$ 11.8
	Swish	911 $\pm$ 15.1
	APL	894 $\pm$ 11.5
	Tent	881 $\pm$ 11.1
	SPLASH	870 $\pm$ 12.3
Our proposed ResNet-20	ReLU	665 $\pm$ 7.9

Bayesian neural networks. We seek the α hyperparameter to control the weights towards adversarial robustness. Two important concepts, i.e, randomness and ensemble, are combined in our approach to prevent strong gradient-based attacks. Multiple inferences with randomness and ensemble are the reason for the high robustness.

3.4.2 Bayes without Bayesian Learning

In this section, we introduce Bayes without Bayesian Learning algorithm applied for the BNN inference to improve the robustness of pre-trained DNNs.

Let x be an original image and y be a corresponding label in a dataset D. We use l_∞ norm PGD attack and l_2 norm C&W attack to generate an adversarial image $\hat{x}$ and employ pre-trained DNN models with learned parameters θ to construct BNNs. The inference process of BNN via BwoBL method is listed in Algorithm 1.

From Eq. (13), we can see that a large α leads to a big variance of $\mathbf{w}$ that causes the uncertainty on model parameters. This uncertainty is beneficial in the resistance to gradient-based perturbation. In our experiment, α depends on pre-trained DNN architectures, it is thus called *structural hyperparameter*. This hyperparameter is chosen and fixed for each pre-trained DNN before the execution of the inference phase. Based on the testing accuracy of our BNN model under adversarial attacks, we select fixed values of α that are explained in each experiment.

Algorithm 1. The inference phase of BNNs with BwoBL algorithm

Input: dataset D and learned parameters of pre-trained DNNs θ
Initialize: $\xi \sim \mathcal{N}(0, I)$ and α
Weights of Bayes layers: $\mathbf{w} = \theta + \alpha\theta \odot \xi$
for (x, y) **in** D **do**
  $\hat{x} =$ *attack_function* (θ, x, y)
  for $i = 1, 2, ...$ **do** # n_ensemble
    $\hat{y}_i \leftarrow f_\xi(\mathbf{w}, \hat{x})$ # several forward passes
  end for
  $\hat{y} =$ *majority_voting*$(\hat{y}_i)$ # the most frequent output
end for

In general, the solution to the parameters of the BNN model cannot be revealed by a closed-form expression because of the probabilistic variable $\xi \sim \mathcal{N}(0, I)$. Consequently, Monte Carlo sampling is utilized to take one sample for each probabilistic variable by random number generation. If a pre-trained DNN is

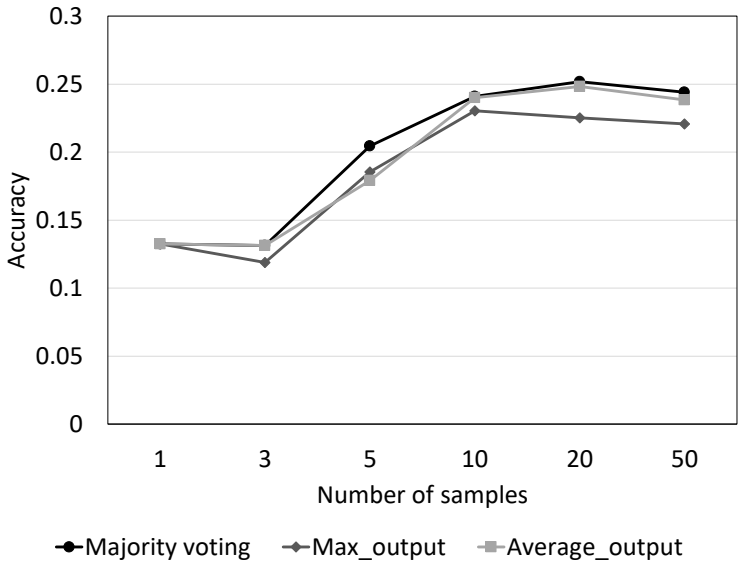

Figure 15. Comparison between the majority voting output, the average output, and the maximum output of the ensemble phase.

$f(w, x)$ with a parameter w and an input x, our BNN can be denoted as $f_\xi(w, x)$ that is an infinite number of stochastic models. Therefore, we can execute the probabilistic model multiple times, which is the ensemble inference procedure, to improve the accuracy of our models.

The ensemble is a beneficial property of BNN that can raise the performance of the model. Due to the model uncertainty, we execute several forward propagations in the testing phase and achieve a set of different outputs. Then, how to take an output of the ensemble model? In order to answer this inquiry, we conduct three ways: (1) the output with the most frequency called the majority voting, (2) the average output, and (3) the maximum output. As an example, we implement ResNet-50 with $\alpha = 0.3$ on ImageNet under l_∞ norm PGD attack at $\epsilon = 2/255$. Fig. 15 confirms that majority voting is the best stable and efficient output. Hence, we use the majority voting output to evaluate the experiments. Moreover, we can see that a large number of samples plays an important role in ensemble inference. It brings better accuracy but costs a considerable amount of inference time. We make a comparison of inference time between adversarial training and our method in Table 5. Our proposal does not consume the training time as

Table 5. Comparison of inference time between adversarial training and our proposal. Latency is the average inference time for one image. Each experiment is run on a single core of Intel®Core™i7-3970X CPU and a GeForce RTX1080Ti GPU.

Method	**Latency (s)**	**# forward passes**
Fast adversarial training (ResNet-50)	0.0293	1
Our proposal (Bayesian ResNet-50)	0.7524	20

adversarial training but needs a large inference time for the ensemble phase. Accordingly, selecting a number of samples must guarantee high accuracy and reasonable inference time.

4 Robustness of BNNs with BwoBL algorithm

4.1 Assessment on CIFAR-10/100

4.1.1 Setup of BNN models and adversarial attacks

We carry out Bayes without Bayesian Learning algorithm on two small datasets that are CIFAR-10, CIFAR-100. We utilize pre-trained ResNet-18 on natural images [143, 144] to build our Bayesian ResNet-18 (ResNet-18 + proposed BwoBL), as introduced in Section 3.2.1. Additionally, we integrate our algorithm into pre-trained models on adversarial images, which have been obtained by Wong et al. [98], and call it FAST shortly. PreActResNet-18 [98] for CIFAR-10 and CIFAR-100 are used in FAST training method. We call them PreActResNet-18_FAST in our experiments and produce Bayesian PreActResNet-18 (PreActResNet-18_FAST + proposed BwoBL), as shown in Section 3.2.2. All experiments are run on a PC with two GeForce GTX1080Ti.

It is known that Bayesian learning becomes hard for large DNN architectures and high-dimensional data. However, Bayesian Neural Network is able to be an efficient defense against adversarial attacks. Therefore, we proposed BwoBL algorithm to apply the Bayesian approach to DNNs and use learned weights of CNNs instead of Bayesian learning towards resisting PGD attacks.

PGD attacks are referred to be the strong iterative attack, which is a universal adversary among all first-order adversaries. From Eq. (9) we set PGD attack under the pixel perturbation $\epsilon = 8/255$ and the iteration $it = \{20, 50, 100\}$ for CIFAR-10 and CIFAR-100. The clip function in Eq. (9) enables PGD to iterate more steps. Hence, this attack is really complicated to defensive methods, especially adversarial training.

4.1.2 Structural hyperparameter and ensemble

Structural hyperparameter: From Eq. (13), we test the performance of our algorithm under different values of the structural hyperparameter α to adjust the standard deviation σ of Gaussian distribution on weights of convolutional layers in pre-trained ResNet-18 and PreActResNet-18.

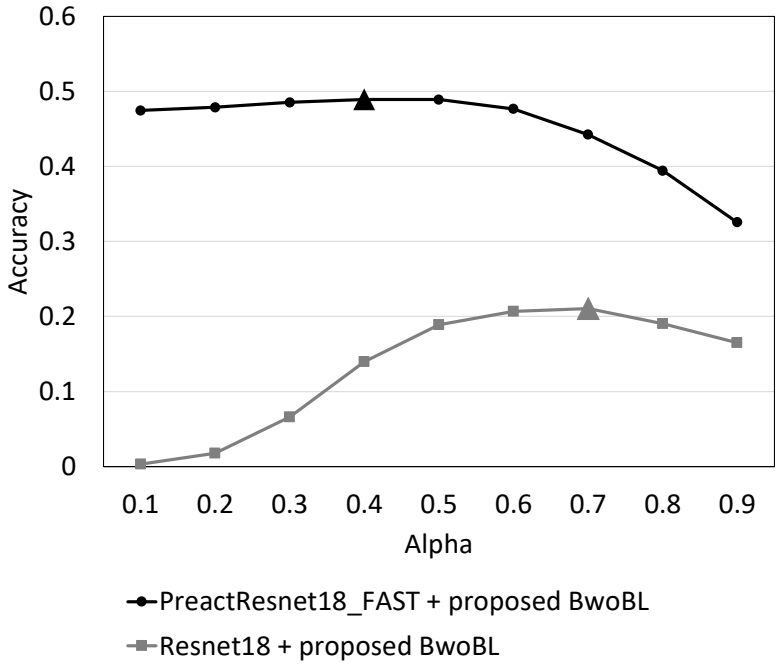

Figure 16. The influence of α on the robustness of proposed models, under PGD attack of $\epsilon = 8/255$ and $it = 20$ on CIFAR-10.

In this experiment, we treat θ of Eq. (13) as learned weights of models and check the effect of the structural hyperparameter on the robustness of the model by changing different α levels. For example, we select

$$\alpha = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\} \tag{20}$$

for ResNet-18 and PreActResNet-18_FAST on CIFAR-10. The result is demonstrated in Fig. 16.

We can see that the best robustness of ResNet-18 and PreActResNet-18_FAST against PGD attack of $\epsilon = 8/255$ and $it = 20$ on CIFAR-10 are at $\alpha = 0.7$ and $\alpha = 0.4$, respectively. These values of the structural hyperparameter are also good for ResNet-18 and PreActResNet-18_FAST on CIFAR-100. We hence fix them for ResNet18 and PreActResNet-18_FAST on both CIFAR-10 and CIFAR-100.

Ensemble: As mentioned in Section 3.4, the ensemble phase is a crux of BNNs, which is executed by several forward propagations and the output of the ensemble inference is majority voting in our experiments.

Many studies have proved the uncertainty on weights of BNNs provides the ensemble model that obtains better performance than the single model. Nonetheless, the difficulty of the ensemble inference is how many samples are enough for execution. For both CIFAR-10 and CIFAR-100, we choose the ensemble of 10 samples that is good enough for the robust assessment of our BNNs against the PGD attack of $\epsilon = 8/255$ and $it = \{20, 50, 100\}$.

4.1.3 Tolerance for adversarial attacks

As mentioned above, the pre-trained ResNet-18 and PreActResNet-18_FAST have been used for applying BwoBL algorithm to build Bayesian ResNet-18 with $\alpha = 0.7$ and Bayesian PreActResNet-18 with $\alpha = 0.4$. This application is against l_∞ norm PGD attack of pixel perturbation $\epsilon = 8/255$ and iteration $it = \{20, 50, 100\}$ on both CIFAR-10 and CIFAR-100 when the values of pixels are in $[0, 255]$.

It is noted that FAST adversarial training [98] is the fastest training method currently that maintains the robust accuracy of CNN model resisting PGD attacks. In this way, Wong et al. [98] referred to FGSM training, which was efficient towards PGD adversaries but had a lower cost. Therefore, we use the results of FAST adversarial training to compare with our BwoBL algorithm. The comparison between conventional CNN, FAST, and our proposal on CIFAR-10 and CIFAR-100 under PGD attacks of $\epsilon = 8/255$ and $it = \{20, 50, 100\}$ steps can be found in Figures 17 and 18.

Our algorithm has considerably improved the testing accuracy of ResNet-18 that are trained on natural images, called *CNN_no defense*. Below strong l_∞ norm PGD attacks, *CNN_no defense* cannot almost distinguish adversarial examples, but our BwoBL can get 25% accuracy on CIFAR-10 without any additional training, as shown in Fig. 17. When we combine BwoBL approach with the pre-trained PreActResNet-18 on perturbed images, the robustness of PreActResNet-18_FAST + BwoBL model is 50%, which is the best among defenses on CIFAR-10 currently.

CIFAR-100 is known to be a more complex dataset with 100 classes. Thus, our algorithm only has the 10% prediction in the top-1 accuracy when it is applied to the pre-trained ResNet-18 on natural examples. 26% accuracy can achieve with the combination of PreActResNet-18_FAST and BwoBL, as seen in Fig. 18 (above). Because of a dataset with more classes, we evaluated the robustness

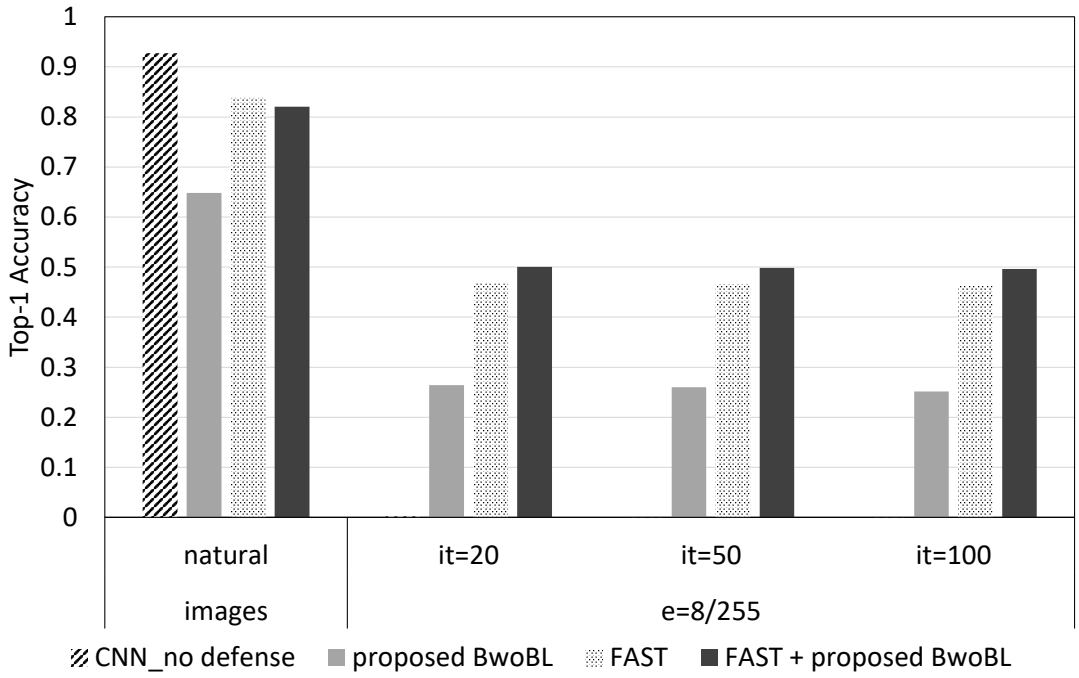

Figure 17. The robustness (top-1 accuracy) of proposed models is compared to naturally pre-trained ResNet-18 (CNN_no defense) and pre-trained PreActResNet-18 on adversarial images (FAST) against PGD attack of $\epsilon = 8/255$ and $it = \{20, 50, 100\}$ on CIFAR-10.

of models in the top-5 accuracy. From Fig. 18 (below), we see that BwoBL algorithm can have the top-5 accuracy that improves FAST adversarial training. For example, with PGD attack of $\epsilon = 8/255$ and $it = 20$, 12% is the top-5 prediction of naturally pre-trained ResNet-18 but the top-5 accuracy of (pre-trained ResNet-18 + proposed BwoBL) is 50% while 57% is the accuracy of (pre-trained PreActResNet-18_FAST + proposed BwoBL). As a result, our BwoBL algorithm can considerably boost the robustness of pre-trained networks but we do not need to re-train these models when attack parameters change.

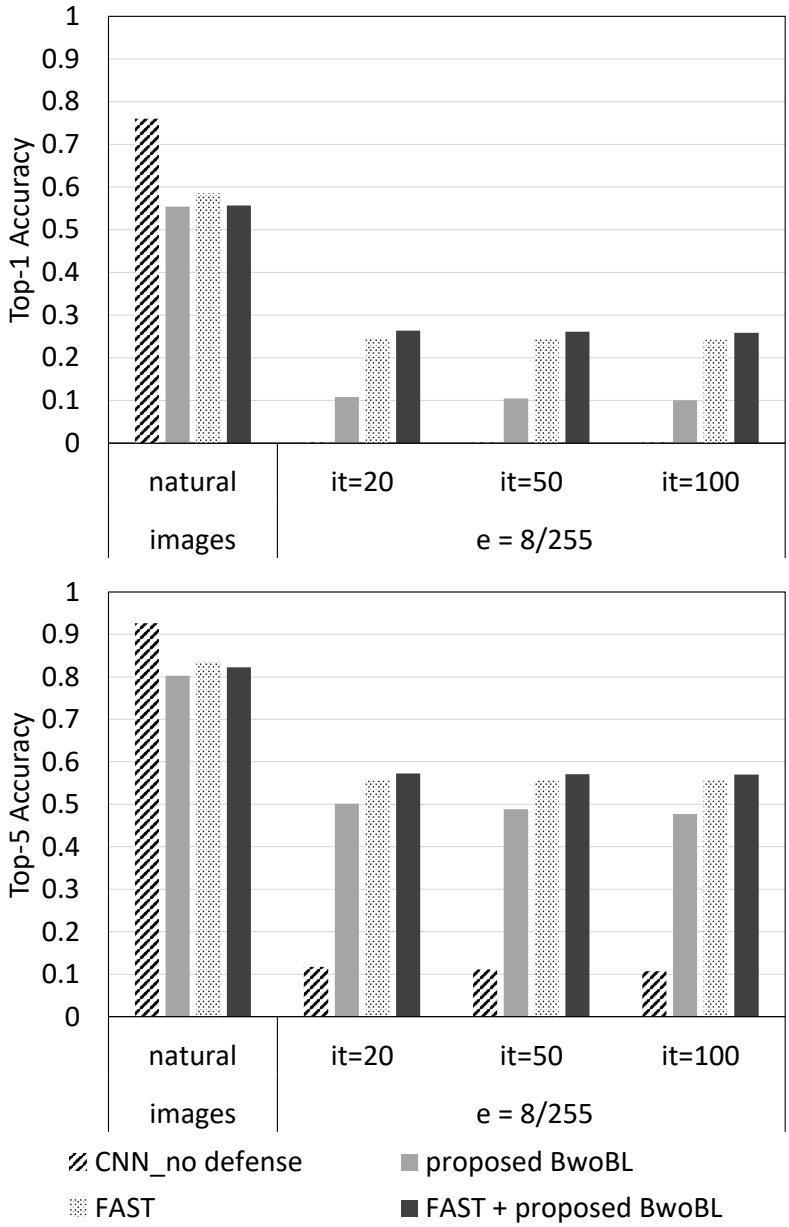

Figure 18. The robustness of proposed models is compared to naturally pre-trained ResNet-18 (CNN_no defense) and pre-trained PreActResNet-18 on adversarial images (FAST) against PGD attack of $\epsilon = 8/255$ and $it = \{20, 50, 100\}$ on CIFAR-100. Above: top-1 accuracy. Below: top-5 accuracy.

4.2 Evaluation on ImageNet

4.2.1 Setup of BNN models

We build up BNN models from the typical DNN architectures currently, which achieve state-of-the-art accuracy on ImageNet. This is the realistic dataset for the task of classifying 1000 classes with large-scale images. The fact that adversarial training on ImageNet considerably consumes the computation time and the training cost. Accordingly, there are only a few findings of adversarial training on ImageNet successfully.

Baseline CNN models used in our experiments are ResNet-50, ResNet-101, ResNet-152, ResNeXt-101-32×8d, EfficientNet-B0∼B7, in which ResNet-50 is the backbone network. ResNet-101 and ResNet-152 are respectively 2× and 3× deeper than ResNet-50, and ResNeXt-101-32×8d is 2× deeper and 4× wider than ResNet-50. These models show that deeper and wider networks can further improve the performance in image recognition. Moreover, Tan et al. [4] argued that scaling up CNNs by their depth or width has not yet achieved the best accuracy and efficiency. They investigated a compound scaling method, which balanced three dimensions of networks, i.e., depth/width/resolution in EfficientNets. These are newly outstanding networks trained on ImageNet. We use all pre-trained EfficientNets on ImageNet [143, 145] to apply our BwoBL algorithm. Our BNNs are built on pre-trained CNNs as mentioned in Section 3.2.

Our algorithm is integrated into not only CNNs but also other powerful DNNs, such as SANs. Recent work has shown effective architectures that employ a self-attention mechanism as an alternative for convolutional networks in image classification. Zhao et al. [115] explored two types of SAN, including pairwise and patchwise self-attention. We replace all linear transformation layers in SANs with our Bayes-linear layers and choose pre-trained SAN19-pairwise and SAN19-patchwise, which are built on ResNet-50 architecture to apply our proposal, as illustrated in Section 3.3.

So as to validate our hypothesis, we compare the performance of BwoBL with adversarial training. Although we have mentioned robust activation functions in our preliminary evaluation (Section 3.4.1), most of them are executed on small datasets and optimized through adversarial training to enhance their robustness

Table 6. State-of-the-art DNN architectures are used in our experiments. We apply BwoBL algorithm to pre-trained DNNs on natural and adversarial ImageNet to construct our BNN models.

Training data	**Pre-trained networks**
Original ImageNet	ResNet-50
	ResNet-101
	ResNet-152
	ResNeXt-101-32×8d
	EfficientNet-Bx
	SAN19-pairwise
	SAN19-patchwise
Adversarial ImageNet	ResNet-50-FAST
	EfficientNet-ADV-Bx

further to strong attacks. Thus, adversarial training is still an outstanding defense. In particular, FAST [98] is the fastest method and maintains the robustness as previous adversarial training, which implements FGSM training against PGD attack on ImageNet with ResNet-50. We hence select this adversarial learning to combine our BwoBL and call them ResNet-50-FAST. Furthermore, Xie et al. [146] have executed adversarial training on ImageNet with EfficientNets, but their target is the enhancement of image recognition models. They treat perturbed examples as additional instances and propose an improved adversarial training scheme - AdvProp, which is trained on both natural and adversarial images. These networks are referred to as EfficientNet-ADV-Bx in our experiments.

All DNN architectures that we perform are summarized in Table 6. Our experiments are run on four GeForce GTX1080Ti and two GeForce RTX3090.

We have already compared our proposal to adversarial Bayesian training under strong PGD attacks on CIFAR-10 in Table 3. So far, we have not found yet the studies of Bayesian learning on natural or adversarial ImageNet with large models to compare in our experiments. To clarify this problem, we have constructed our Bayesian ResNet-50 and tried to train it on natural and adversarial

Table 7. Comparison of training time between adversarial training on CNN, Bayesian learning on natural images, and Bayesian learning on adversarial images. FAST adversarial training [98] has been implemented in 15 epochs to achieve the most robustness. 10 samples are used to execute the ensemble phase in BNNs. Each experiment is run on a single core of Intel®Core™i9-10920X CPU and a GeForce RTX3090 GPU.

Training method	**# epochs)**	**Training time for 1 epoch (hours)**	**Total training time (hours)**
FAST adversarial training (ResNet-50)	15	2.8	42
Bayesian training on natural images (Bayesian ResNet-50)	15 (expected)	50	750 (expected time)
Bayesian training on adversarial images (Bayesian ResNet-50)	15 (expected)	67	1005 (expected time)

ImageNet. The comparison of training time in Table 7 might be the reason why has no findings of Bayesian learning on ImageNet. From Table 7, it is emphasized that Bayesian learning expends a long training time, which is infeasible when implemented on ImageNet. Compared to traditional learning, it requires more time to converge and does not improve on existing techniques.

4.2.2 Setup of adversarial attacks

We set two forceful kinds of adversarial attacks that are PGD and C&W algorithm to verify the robustness of our method.

From Eq. (9), we generate l_∞ norm PGD attacks with the pixel-perturbed size $\epsilon = \{2/255, 4/255\}$, the step size $\beta = 1/255$, and the attack iterations $it = \{10, 50, 100\}$. It is known that increasing the number of attack iterations makes adversarial examples harder and is an obstruction of adversarial training.

With C&W attack, we compare the resistance of the models under the attack that is measured by l_2 norm of perturbed images, binary_search_steps = 4, max_iterations = 1000, initial_constant = 0.1, and confidence = 10. This establishment creates strong adversaries, which are enough to fool baseline networks.

Notice that both PGD and C&W are white-box and non-targeted attacks. However, the white-box manner is set for pre-trained networks in Table 6. Meanwhile, our BNNs are built on these pre-trained models and utilize their learned parameters. It is then considered indirectly white-box settings.

4.2.3 Structural hyperparameter

The crux of our BwoBL algorithm is *structural hyperparameter* α in Eq. (13), which adjusts the variance of Gaussian distribution on the parameters of Bayes layers. For each pre-trained DNN in Table 6, we seek the best α so that the model achieves the highest accuracy under strong l_∞ norm PGD attack of pixel perturbation $\epsilon = 4/255$, iteration $it = 10$.

We test the performance of our proposed networks in Table 6 with a wide range of α values, for example

$$\alpha = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\} \tag{21}$$

From Fig. 19, we realize the accuracy is a practically convex upward with the change of α. The search space of α guarantees the optimal algorithm, and the best α corresponds to the peak of this convex upward.

Fig. 19 shows that the step size 0.1 of α makes a too big difference in the accuracy of the model, which is difficult to identify the peak exactly. We discover the step size 0.01 is good enough to find the peak of convex upwards. Especially,

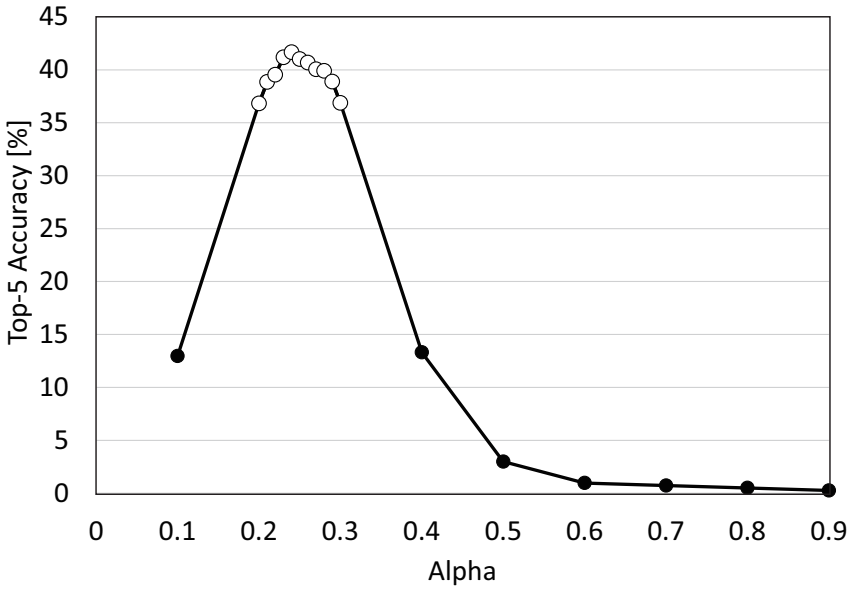

Figure 19. The accuracy is a convex upward with the change of *structural hyperparameter* α. The best α corresponds to the peak of the convex upward. Our network: ResNet-50 + BwoBL. PGD attack: l_∞ norm, $\epsilon = 4/255$, iteration = 10. Black and white markers stand for step size 0.1 and step size 0.01 of α, respectively.

our Bayesian ResNet-50 with $\alpha = 0.2$ and 0.3 get an approximate accuracy, but with

$$\alpha = \{0.21, 0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29\} \tag{22}$$

our proposed model continues increasing the accuracy, and we achieve its peak at $\alpha = 0.24$.

We hereby observe the effect of this factor in our proposed BNNs and treat it as a constant for each model, which is summed up in Table 8. In the proposed approach, we do not execute the additional training phase but we need to seek the α hyperparameter, which controls the weights of a Bayesian neural network towards the adversarial robustness. This search process expends an amount of time for the loops. For example, to achieve a convex upward of the accuracy in Fig. 19, we must perform the search loop for our Bayesian ResNet-50 in 11 hours to determine the best α. Nonetheless, compared to the cost of adversarial training in Table 7, this seeking is insignificant and easy to apply for various models.

Table 8. *Structural hyperparameter* α that adjusts the variance of Gaussian distribution on parameters of Bayes layers fixed for our proposed BNNs.

Architectures	α	**Architectures**	α
ResNet-50	0.24	SAN19-pairwise	0.10
ResNet-101	0.25	SAN19-patchwise	0.15
ResNet-152	0.26	ResNet-50-FAST	0.10
ResNeXt-101-32×8d	0.23		
EfficientNet-B0	0.10	EfficientNet-ADV-B0	0.05
EfficientNet-B1	0.12	EfficientNet-ADV-B1	0.07
EfficientNet-B2	0.12	EfficientNet-ADV-B2	0.08
EfficientNet-B3	0.14	EfficientNet-ADV-B3	0.11
EfficientNet-B4	0.18	EfficientNet-ADV-B4	0.15
EfficientNet-B5	0.22	EfficientNet-ADV-B5	0.17
EfficientNet-B6	0.23	EfficientNet-ADV-B6	0.20
EfficientNet-B7	0.24	EfficientNet-ADV-B7	0.21

4.2.4 Ensemble inference

The ensemble is a beneficial property of Bayesian inference by executing the stochastic model many times. As mentioned in Algorithm 1, we carry out several forward passes and take the ensemble of different output samples. We assess the majority voting output for our ensemble inference.

Consequently, how many samples do we need to implement in the ensemble phase? In real-world applications, ensemble inference is a big challenge because of the longer execution time. A large quantity of samples brings a better performance of the networks but considerably consumes the inference time, especially on big datasets and deeper neural networks. Selecting a number of forward passes must be suitable to achieve high accuracy and reasonable inference time. Based on the achievement in Fig. 20, we are aware that the accuracy differential between the ensemble of 20, 30, 40, and 50 samples is not too great. We thus choose 20 that is tolerable for all our proposed models on ImageNet to obtain a trade-off between the accuracy and the computation cost.

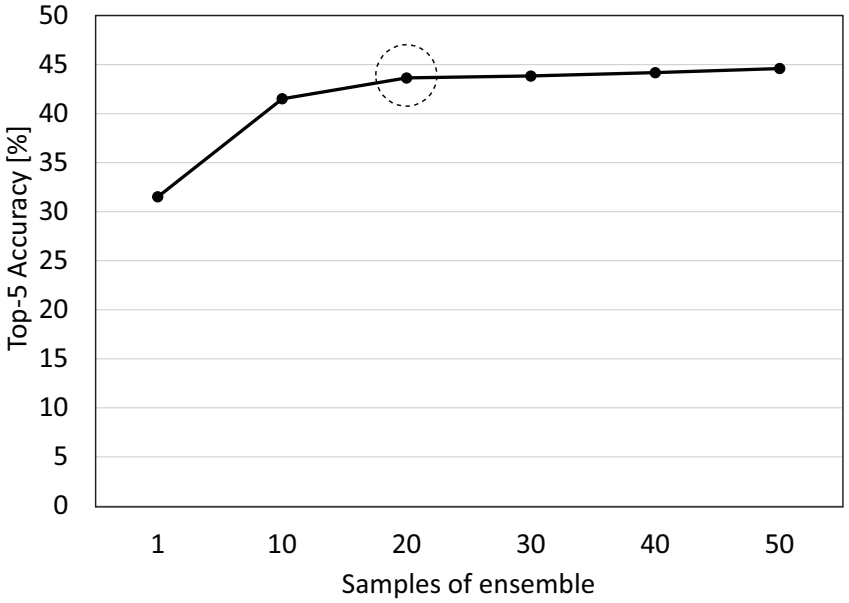

Figure 20. A number of forward passes in ensemble inference. The ensemble of 20 samples is chosen to attain a trade-off between the accuracy and the computation cost for all our proposed models. Our network: ResNet-50 + BwoBL. PGD attack: l_∞ norm, $\epsilon = 4/255$, iteration = 10.

It is apparent that the inference time of BNN is more massive than that of CNN due to the ensemble phase. This repeated inference leads to much longer computation time, which makes BNN difficult for applying to real-world systems [147, 148, 149, 150]. However, Wan et al. [147] succeeded in the design of an FPGA-based hardware accelerator for BNNs, which executed the repeated inference of BNNs but still got 2.1 $\sim$ 8.2$\times$ speedup over the CNN accelerator. This proved that the ensemble inference of BNN could completely be spread to real-world applications to improve the performance of DNNs.

4.2.5 Performance on natural ImageNet

We assess the performance of all pre-trained DNNs and our proposed models on natural ImageNet with 50000 images in the validation set. On large-scale ImageNet with 1000 classes, the top-5 evaluation might achieve the better prediction that we observe in our experiments. Our method mainly focuses on improving the robustness of classifiers via the uncertainty generation on the weights, which makes our BNNs robust to the adversaries rather than natural images. Therefore, applying the BwoBL algorithm to pre-trained DNNs affects the testing accuracy on natural examples, which is a 6.90% drop on average, as seen in Table 9. Many studies of adversarial training indicated that the trade-off between adversarial robustness and standard accuracy was inevitable. The fact that gaining robustness of the model would be losing accuracy on natural examples. We hence apply the BwoBL algorithm to deeper and wider networks to yield comparable accuracies on natural images. Table 9 shows that scaling up networks enhances not only the accuracy of the original DNNs but also that of our proposed networks. In particular, the application of our proposal to scaling networks significantly boosts the accuracy and 95.51%, which is only a 1.72% drop, top-5 prediction of our proposed EfficientNet-ADV-B7 is the advanced accuracy on natural ImageNet among defense methods currently.

4.2.6 Tolerance on PGD adversaries

We create PGD attacks on the first 10000 images in the validation set of ImageNet and evaluate the tolerance of our BwoBL algorithm on these perturbed images.

We examine the robustness of our proposed models under l_∞ norm PGD attack of pixel perturbation $\epsilon = \{2/255, 4/255\}$ and iterations $it = \{10, 50, 100\}$, as shown in Tables 10 and 11. The combination of our BwoBL and pre-trained DNNs on natural images significantly enhances the robustness, which is 44.83% and 60.16% increases on average in ResNets and SANs, respectively. However, it is just a slight improvement when applying BwoBL to pre-trained DNNs on adversarial images, i.e., ResNet-50-FAST, which rises by 3% with pixel perturbation $\epsilon = 4/255$.

Table 9. Comparable top-5 accuracies (%) between pre-trained DNNs and our proposed networks on natural ImageNet.

	Pre-trained networks	**Our proposal**
ResNet-50	92.93	81.75
ResNet-101	93.55	82.75
ResNet-152	94.05	83.42
ResNeXt-101-32×8d	94.53	87.31
EfficientNet-B0	91.04	75.25
EfficientNet-B1	91.71	75.53
EfficientNet-B2	94.39	83.06
EfficientNet-B3	95.31	85.42
EfficientNet-B4	96.12	89.17
EfficientNet-B5	96.62	89.40
EfficientNet-B6	96.90	90.59
EfficientNet-B7	96.94	91.17
SAN19-pairwise	93.36	84.64
SAN19-patchwise	93.90	89.12
ResNet-50-FAST-2px	81.69	81.16
ResNet-50-FAST-4px	77.13	76.55
EfficientNet-ADV-B0	89.19	79.17
EfficientNet-ADV-B1	93.22	87.17
EfficientNet-ADV-B2	94.46	90.74
EfficientNet-ADV-B3	95.10	91.95
EfficientNet-ADV-B4	95.90	93.06
EfficientNet-ADV-B5	96.81	94.51
EfficientNet-ADV-B6	97.16	95.34
EfficientNet-ADV-B7	97.23	95.51

Table 10. Robustness to PGD attacks: l_∞ norm, pixel perturbation $\epsilon = \{2/255, 4/255\}$, iteration $it = \{10, 50, 100\}$, are evaluated by top-5 accuracies (%) and compared between naturally pre-trained DNNs and our proposed networks.

	Pre-trained networks				**Our proposal**			
	$it = 10$		$it = 50$	$it = 100$	$it = 10$		$it = 50$	$it = 100$
	$\epsilon = 2/255$	$\epsilon = 4/255$	$\epsilon = 4/255$	$\epsilon = 4/255$	$\epsilon = 2/255$	$\epsilon = 4/255$	$\epsilon = 4/255$	$\epsilon = 4/255$
ResNet-50	10.44	5.06	2.73	2.15	59.61	42.87	44.72	45.32
ResNet-101	12.15	5.90	3.78	3.21	64.69	50.64	52.49	52.82
ResNet-152	13.50	6.39	3.89	3.51	66.89	53.55	54.51	56.60
ResNeXt-101-32×8d	22.62	15.09	12.07	10.85	75.43	64.70	66.45	67.22
EfficientNet-B0	3.56	0.52	0.09	0.08	45.87	25.41	29.52	31.85
EfficientNet-B1	2.02	0.22	0.07	0.05	42.17	24.98	28.36	31.49
EfficientNet-B2	3.19	0.56	0.18	0.18	61.39	43.58	48.65	52.12
EfficientNet-B3	2.83	0.72	0.37	0.35	67.64	51.08	57.48	60.21
EfficientNet-B4	2.56	0.70	0.45	0.43	76.55	63.73	69.71	72.20
EfficientNet-B5	3.11	0.78	0.23	0.12	78.65	68.82	74.37	76.06
EfficientNet-B6	1.97	0.47	0.13	0.08	78.89	68.36	74.68	76.65
EfficientNet-B7	1.62	0.27	0.06	0.02	**81.20**	**70.88**	**77.84**	**81.41**
SAN19-pairwise	6.82	2.12	0.46	0.22	70.76	59.16	62.59	63.16
SAN19-patchwise	1.03	0.28	0.16	0.14	76.10	63.55	66.94	68.73

Table 11. Robustness to PGD attacks: l_∞ norm, pixel perturbation $\epsilon = \{2/255, 4/255\}$, iteration $it = \{10, 50, 100\}$, are evaluated by top-5 accuracies (%) and compared between adversarial pre-trained DNNs and our proposed networks.

	Pre-trained networks				**Our proposal**			
	$it = 10$		$it = 50$	$it = 100$	$it = 10$		$it = 50$	$it = 100$
	$\epsilon = 2/255$	$\epsilon = 4/255$	$\epsilon = 4/255$	$\epsilon = 4/255$	$\epsilon = 2/255$	$\epsilon = 4/255$	$\epsilon = 4/255$	$\epsilon = 4/255$
ResNet-50-FAST-2px	70.16	56.61	56.49	56.40	71.88	59.52	59.82	59.84
ResNet-50-FAST-4px	66.95	57.83	57.41	56.31	68.67	60.26	59.50	58.59
EfficientNet-ADV-B0	42.42	12.72	8.31	8.26	62.53	45.62	48.76	48.78
EfficientNet-ADV-B1	53.90	18.11	10.29	10.39	76.55	63.58	64.81	65.24
EfficientNet-ADV-B2	55.52	18.90	10.34	9.25	82.64	73.42	75.70	75.82
EfficientNet-ADV-B3	65.52	30.36	19.55	17.94	86.69	80.50	81.91	81.91
EfficientNet-ADV-B4	64.57	30.53	16.61	15.32	88.52	85.16	85.80	86.22
EfficientNet-ADV-B5	66.50	32.97	15.49	13.41	91.86	89.29	89.90	90.23
EfficientNet-ADV-B6	64.90	31.51	12.33	10.89	93.55	90.92	91.40	91.57
EfficientNet-ADV-B7	62.07	30.30	9.17	7.82	**93.72**	**91.48**	**92.05**	**92.14**

SAN19 is built on ResNet-50 architecture, but its parameters and FLOPs are smaller and its accuracy is higher than ResNet-50 on natural images. The application of the BwoBL algorithm to SAN19 sharply raises the robustness of naturally pre-trained SAN19-pairwise and SAN19-patchwise (Table 10). In which the top-5 accuracies of our SAN19-patchwise are 76.10% and 63.55% on 2-pixel and 4-pixel perturbed images, respectively, which outperform ResNet-50. This proves that the BwoBL approach may be integrated into any DNN architecture to construct a robust network against adversaries.

Table 10 also demonstrates that scaling up ResNet by the depth and the width can considerably boost the resistance of our algorithm to adversarial attacks. For instance, under the $\epsilon = 4/255$ and $it = 10$ PGD attack, our ResNeXt-101-32×8d, which is built on naturally pre-trained ResNeXt-101-32×8d, improves the robustness by 21.83% for our ResNet-50, 4.44% for our ResNet-50-FAST-4px, and 5.18% for our ResNet-50-FAST-2px. Furthermore, adversarial training must be iterated to achieve the highest accuracy when attack parameters change, such as ResNet-50-FAST-2px and ResNet-50-FAST-4px are ResNet-50 trained on 2-pixel and 4-pixel perturbed images (Table 11). In contrast, our algorithm is conveniently applied to those models to resist forceful PGD attacks without an additional training phase.

Additionally, increasing the iteration generates stronger PGD attacks but it also raises the training cost. Thus, adversarial training usually uses single-step PGD as FGSM to lower the training cost. Nonetheless, the robustness of the models trained on FGSM examples that is ResNet-50-FAST-2px and ResNet-50-FAST-4px will decline when the number of PGD iterations rises, as seen in Table 11. Contrarily, our proposed models are more robust with powerful attacks.

EfficientNet is a strong family of DNNs for image classification, which determines the optimal factor for scaling networks. We observe the benefit of EfficientNet in our proposed algorithm with two pre-trained EfficientNet schemes. Our EfficientNet is the connection between the BwoBL method and pre-trained EfficientNet on natural images. The proposed EfficientNet-ADV is applying BwoBL to pre-trained EfficientNet on both natural and PGD adversarial images.

We evaluate the robustness of our BwoBL method on EfficientNets under l_∞ PGD perturbation of pixels $\epsilon = \{2/255, 4/255\}$ with iterations $it = \{10, 50, 100\}$.

As shown in Table 10, pre-trained EfficientNets on natural images are almost fooled by the attacks but our proposed EfficientNets remarkably boost the accuracy without the extra training. For example, the top-5 accuracy of EfficientNet-B7 rises from 0.02% to 81.41% below PGD attack of $\epsilon = 4/255, it = 100$. Furthermore, Table 11 prove that EfficientNet-ADV has just resisted weak attacks like $\epsilon = 2/255, it = 10$ and becomes unsteady with respect to intense attacks. Meanwhile, our EfficientNet-ADV is always robust against any attacks. Particularly, the proposed EfficientNet-ADV-B7 achieves the cutting-edge accuracy under both $\epsilon = 2/255, it = 10$ and $4/255, it = 100$ perturbation, which is 93.72% and 92.14%, while the accuracies of pre-trained EfficientNet-ADV-B7 are only 62.07% and 7.82%, respectively. In our understanding, 92.14% top-5 accuracy of our EfficientNet-ADV-B7 is currently the best robustness of DNN under strong PGD attack of $\epsilon = 4/255, it = 100$ among defense methods.

From Tables 10 and 11, it is emphasized that our proposal is stably robust on any networks against any attacks, although pre-trained models are trained on natural or adversarial images.

4.2.7 Tolerance on C&W adversaries

C&W attack is evaluated as an additional adversary to verify the robustness of our BwoBL algorithm. The convergence of C&W is slower than that of PGD. It hence expends considerable time to generate a C&W attack. In the original paper of the C&W attack [31], Carlini et al. only assessed the first 1000 images in the test set of CIFAR-10 and MNIST. For that reason, we figure out the performance of all networks on the first 1000 images in the validation set of ImageNet.

Even though we determine *structural hyperparameter* α of the weights to build our BNNs based on PGD attack, our proposed models are still robust under strong C&W attack, as shown in Table 12. We realize ResNet architectures are powerful with regard to C&W attacks, and our algorithm strengthens their performance more, i.e., a 28.6% rise on average. Other models, such as EfficientNets and SANs, are easily attacked, but if they are combined with our BwoBL, their robustness is substantially reinforced. From the recognition of baseline pre-trained networks is less than 10%, our proposed EfficientNets and SANs are able to reach 88.5% and 84.4% maximum top-5 accuracy under strong C&W attack without any additional

training phase.

It is pointed out that adversarial training with a PGD attack cannot resist C&W attacks. In Table 12, ResNet-50-FAST-2px and ResNet-50-FAST-4px, which are trained with PGD adversarial images only achieve 12.2% and 9.3% on C&W attack images. Moreover, when applying our algorithm to both of them, their top-5 accuracies increase by 4.4% and 6.0%, respectively. It means the application of our BwoBL to naturally pre-trained networks is more robust than the application to pre-trained models on adversarial examples. However, pre-trained DNNs on the mixed dataset, including natural and PGD adversarial images, as EfficientNet-ADV might become the best defense against any attacks when they are combined with our BwoBL algorithm. As seen in Table 12, EfficientNet-ADV models are almost fooled by the C&W attack but our EfficientNet-ADV networks considerably enhance the robustness. 94.2% of the proposed EfficientNet-ADV-B7 is the cutting-edge accuracy currently under strong C&W attack.

Tables 10, 11, and 12 verify the best robustness of our proposed BNNs resisting both PGD and C&W attacks, though we apply the BwoBL algorithm to naturally pre-trained DNNs or adversarial pre-trained models.

Table 12. Robustness of our proposed networks and pre-trained DNNs to l_2 norm C&W attack are assessed by top-5 accuracies (%).

	Pre-trained networks	**Our proposal**
ResNet-50	42.5	76.9
ResNet-101	47.1	77.1
ResNet-152	46.0	79.0
ResNeXt-101-32×8d	67.4	84.3
EfficientNet-B0	3.0	60.7
EfficientNet-B1	1.3	56.9
EfficientNet-B2	1.7	73.3
EfficientNet-B3	2.2	81.8
EfficientNet-B4	2.2	83.6
EfficientNet-B5	1.2	85.8
EfficientNet-B6	0.7	85.9
EfficientNet-B7	0.3	88.5
SAN19-pairwise	3.1	78.6
SAN19-patchwise	6.5	84.4
ResNet-50-FAST-2px	12.2	16.6
ResNet-50-FAST-4px	9.3	15.3
EfficientNet-ADV-B0	4.0	53.4
EfficientNet-ADV-B1	4.3	71.1
EfficientNet-ADV-B2	5.3	76.3
EfficientNet-ADV-B3	7.6	82.5
EfficientNet-ADV-B4	7.6	88.8
EfficientNet-ADV-B5	3.9	90.3
EfficientNet-ADV-B6	4.0	91.8
EfficientNet-ADV-B7	1.9	**94.2**

4.2.8 Comprehensive assessment

To sum up, our BwoBL approach can construct BNNs based on learned parameters of DNNs and sharply improves the defense of pre-trained DNNs against any adversarial attacks but does not need iteratively adversarial training. Fig. 21 summarizes a comparison between the robustness of our proposal, standard and adversarial training. Standard training often gets a good accuracy on natural images and becomes a valuable defense when it is combined with our proposal. In addition, PGD adversarial training is just good for PGD attacks but not against C&W attacks, which reduces the accuracy of standard training, as shown in ResNet-50 of Fig. 21. Otherwise, adversarial training in EfficientNets is not only gaining accuracy but also gaining robustness with applying our proposed algorithm. Fig. 21 indicates that our BwoBL method importantly contributes to improving the robustness of pre-trained models with major natural data and cheaper adversaries resisting a diversity of adversarial attacks.

In reality, security-critical applications need a robust technique as Bayes without Bayesian Learning that does not cost additional training computation and is independent of attack algorithms but is expected to resist any attack in the future. With these observations, we prove that Bayes without Bayesian Learning algorithm significantly boosts the robustness of pre-trained models against adversarial attacks on real datasets and deep neural networks that are challenging to the training problem.

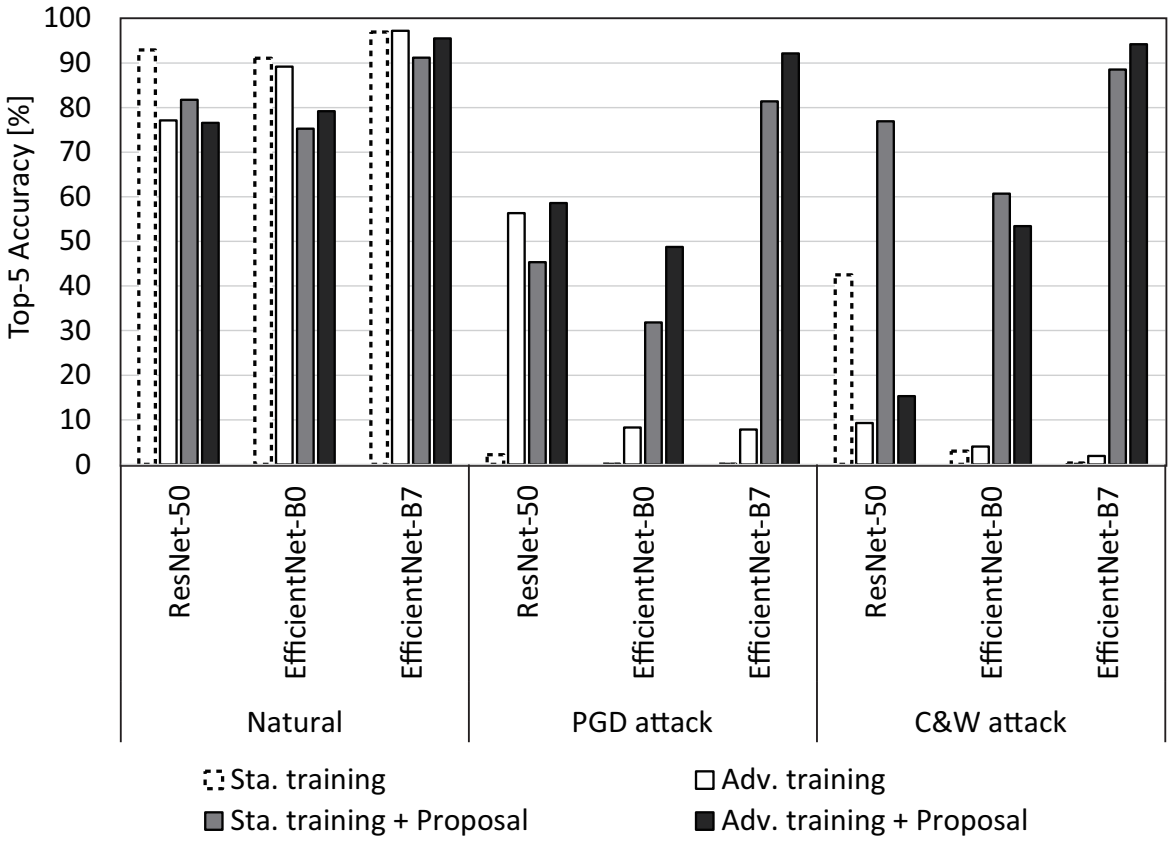

Figure 21. Summarized comparison between our proposal and other defense methods of typical networks on natural images, PGD attack images: l_∞ norm, $\epsilon = 4/255$, iteration = 100, and C&W attack images.
Sta. and Adv. training stand for Standard and Adversarial training.

5 Conclusion

In this dissertation, we build various BNNs and explore a robust Bayesian inference against adversarial attacks, which has been based on Bayes without Bayesian Learning algorithm. We mentioned that most of the previous defenses performed adversarial training as an outstanding method. As we presented, the cost of training and generating adversarial examples are substantial in adversarial learning. Additionally, catastrophic overfitting is a difficulty in this training process. These reasons make adversarial training more complicated on large-scale data under strong attacks. So as to avoid the iterative training towards a variety of perturbations, we focus on Bayesian inference from pre-trained DNNs. Our method builds BNNs on the state-of-the-art pre-trained DNNs via replacing convolutional layers of pre-trained CNNs and linear transformation layers of pre-trained SANs with our Bayes layers that is an application of transfer learning. By treating the mean of BNN weights as single-point values of the weights of pre-trained DNNs, we concentrate on controlling the variance of the probabilistic distribution of BNN parameters, which is assumed a structural hyperparameter. Each pre-trained DNN is fixed by a structural hyperparameter to produce the weight uncertainty resisting gradient-based attacks. Furthermore, the ensemble model that is generated by the random sampling of probabilistic weights enhances the performance of our BNNs. We hereby declare Bayesian inference as an effective resistance to a variety of adversaries without costing adversarial Bayesian learning. Our Bayesian inference method reveals its potential towards adversarial robustness on both small and large data, such as CIFAR-10/100 and ImageNet. Especially, under l_∞ norm PGD attack of pixel perturbation $\epsilon = 4/255$ with 100 iterations and strong l_2 norm C&W attack on the real dataset like ImageNet, the top-5 accuracies of our proposed models increase by 58.18% and 62.26% on average, respectively, which are combined with naturally pre-trained networks. The most powerful model, i.e., our EfficientNet-ADV-B7, achieves 92.14% and 94.20% accuracy under these intense PGD and C&W attacks, which are the best robustness among defense methods recently. In the future, we hope that this inference approach can be widely applied in multiple DNN architectures and become a robust technique of deep learning security towards a diversity of adversaries.

Acknowledgements

First of all, I would like to sincerely thank my Ph.D. supervisor, Prof. Yasuhiko Nakashima, for his enthusiastic guidance and continuous support over the last three years. Under the supervision of Prof. Nakashima, I began the interests in deep learning security, the limitations of deep neural networks, which inspired me to explore a new research direction - Bayesian Neural Networks resisting adversarial attacks. This research is throughout my Ph.D. course, in which I have published some studied work. Besides, I would also like to thank Prof. Kazushi Ikeda, Assoc. Prof. Renyuan Zhang, and Visiting Assoc. Prof. Tran Thi Hong for serving on my thesis committee and giving valuable suggestions. Likewise, I deeply appreciate the mentoring of Assoc. Prof. Takashi Nakada, who directly guide me when I am a newbie researcher in the field of deep learning. He has taught me how to think critically and explore many aspects of the research, and kept supporting me even after he left our institute.

Next, I would like to thank Computing Architecture Laboratory, Division of Information Science, Nara Institute of Science and Technology (NAIST) for supporting my academic career. I feel so lucky to get a chance to work with a lot of wonderful colleagues at Computing Architecture Laboratory in the last three years. Additionally, I feel happy when spending my life with a small Vietnamese community at NAIST during that time. I would like to thank all people who aid me to overcome the difficult and boring life during the Covid-19 pandemic in Japan.

Last but not least, I would like to express my gratitude to my family members. I am grateful to my parents and mother-in-law for unconditionally supporting me so that I can pursue my academic dream. I am thankful to my husband and son for always standing by my side at all times even though the geographical distance is our obstruction.

References

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[2] [Online]. Bayes by backprop from scratch. Available: https://gluon.mxnet.io/chapter18-variational-methods-and-uncertainty/bayes-by-backprop.html.

[3] Kumar Shridhar, Felix Laumann, and Marcus Liwicki. A comprehensive guide to bayesian convolutional neural network with variational inference. *arXiv preprint arXiv:1901.02731*, 2019.

[4] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[5] Mohammadamin Tavakoli, Forest Agostinelli, and Pierre Baldi. Splash: Learnable activation functions for improving accuracy and adversarial robustness. *Neural Networks*, 140:1–12, 2021.

[6] Clarence Chio and David Freeman. *Machine learning and security: Protecting systems with data and algorithms.* " O'Reilly Media, Inc.", 2018.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[8] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1915–1929, 2012.

[9] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[10] Constantine P Papageorgiou, Michael Oren, and Tomaso Poggio. A general framework for object detection. In *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, pages 555–562. IEEE, 1998.

[11] Constantine Papageorgiou and Tomaso Poggio. A trainable system for object detection. *International journal of computer vision*, 38(1):15–33, 2000.

[12] Paul Viola, Michael Jones, et al. Robust real-time object detection. *International journal of computer vision*, 4(34-47):4, 2001.

[13] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537, 2011.

[14] Gobinda G Chowdhury. Natural language processing. *Annual review of information science and technology*, 37(1):51–89, 2003.

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[16] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018.

[17] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR-2015, arXiv preprint arXiv:1409.1556*, 2014.

[18] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[19] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[20] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[21] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *3rd International Conference on Learning Representations - ICLR 2015, arXiv preprint arXiv:1412.6572*, 2014.

[22] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

[23] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *2nd International Conference on Learning Representations - ICLR 2014, arXiv preprint arXiv:1312.6199*, 2013.

[24] Alexey Kurakin, Ian Goodfellow, Samy Bengio, et al. Adversarial examples in the physical world, 2016.

[25] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015.

[26] Ali Rahmati, Seyed-Mohsen Moosavi-Dezfooli, Pascal Frossard, and Huaiyu Dai. Geoda: a geometric framework for black-box adversarial attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8446–8455, 2020.

[27] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1625–1634, 2018.

[28] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773, 2017.

[29] Hyun Kwon, Yongchul Kim, Ki-Woong Park, Hyunsoo Yoon, and Daeseon Choi. Advanced ensemble adversarial example on unknown deep neural network classifiers. *IEICE TRANSACTIONS on Information and Systems*, 101(10):2485–2500, 2018.

[30] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *6th International Conference on Learning Representations - ICLR 2018, arXiv preprint arXiv:1706.06083*, 2017.

[31] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[32] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

[33] [Online]. Adversarial example generation. Available: https://pytorch.org/tutorials/beginner/fgsm_tutorial.html.

[34] Alexey Kurakin, Ian Goodfellow, Samy Bengio, Yinpeng Dong, Fangzhou Liao, Ming Liang, Tianyu Pang, Jun Zhu, Xiaolin Hu, Cihang Xie, et al. Adversarial attacks and defences competition. In *The NIPS'17 Competition: Building Intelligent Systems*, pages 195–231. Springer, 2018.

[35] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1778–1787, 2018.

[36] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016.

[37] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. Efficient defenses against adversarial attacks. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 39–49, 2017.

[38] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 501–509, 2019.

[39] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 3–14, 2017.

[40] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

[41] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *Network and Distributed Systems Security Symposium 2018, arXiv preprint arXiv:1704.01155*, 2017.

[42] Guneet S Dhillon, Kamyar Azizzadenesheli, Zachary C Lipton, Jeremy Bernstein, Jean Kossaifi, Aran Khanna, and Anima Anandkumar. Stochastic activation pruning for robust adversarial defense. *ICLR 2018, arXiv preprint arXiv:1803.01442*, 2018.

[43] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. *32nd Conference on Neural Information Processing Systems (NIPS 2018)*, 2018.

[44] Andras Rozsa and Terrance E Boult. Improved adversarial robustness by reducing open space risk via tent activations. *arXiv preprint arXiv:1908.02435*, 2019.

[45] Bao Wang, Alex Lin, Penghang Yin, Wei Zhu, Andrea L Bertozzi, and Stanley J Osher. Adversarial defense via the data-dependent activation, total variation minimization, and adversarial training. *Inverse Problems & Imaging*, 2020.

[46] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, 2019.

[47] Ali Shafahi, Mahyar Najibi, Zheng Xu, John Dickerson, Larry S Davis, and Tom Goldstein. Universal adversarial training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5636–5643, 2020.

[48] Thilo Strauss, Markus Hanselmann, Andrej Junginger, and Holger Ulmer. Ensemble methods as a defense to adversarial perturbations against deep neural networks. *arXiv preprint arXiv:1709.03423*, 2017.

[49] Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. Towards robust neural networks via random self-ensemble. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 369–385, 2018.

[50] Xuanqing Liu, Yao Li, Chongruo Wu, and Cho-Jui Hsieh. Adv-bnn: Improved adversarial defense through robust bayesian neural network. *ICLR 2019, arXiv preprint arXiv:1810.01279*, 2018.

[51] Christopher M Bishop. Bayesian methods for neural networks. 1995.

[52] David JC MacKay. Bayesian methods for backpropagation networks. In *Models of neural networks III*, pages 211–254. Springer, 1996.

[53] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[54] José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International conference on machine learning*, pages 1861–1869. PMLR, 2015.

[55] David Barber and Christopher M Bishop. Ensemble learning in bayesian neural networks. *Nato ASI Series F Computer and Systems Sciences*, 168:215–238, 1998.

[56] Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust bayesian neural networks. *Advances in neural information processing systems*, 29:4134–4142, 2016.

[57] William D Penny and Stephen J Roberts. Bayesian neural networks for classification: how useful is the evidence framework? *Neural networks*, 12(6):877–892, 1999.

[58] Shengyang Sun, Changyou Chen, and Lawrence Carin. Learning structured weight uncertainty in bayesian neural networks. In *Artificial Intelligence and Statistics*, pages 1283–1292. PMLR, 2017.

[59] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[60] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[61] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[62] [Online]. Imagenet database. Available: https://image-net.org/index.php.

[63] Thi Thu Thao Khong, Takashi Nakada, and Yasuhiko Nakashima. Bayes without bayesian learning for resisting adversarial attacks. In *2020 Eighth International Symposium on Computing and Networking (CANDAR)*, pages 221–227. IEEE, 2020.

[64] Thi Thu Thao Khong, Takashi Nakada, and Yasuhiko Nakashima. Flexible bayesian inference by weight transfer for robust deep neural networks. *IEICE Transactions on Information and Systems*, E104-D(11):1981–1991, 2021.

[65] Roland Orre, Anders Lansner, Andrew Bate, and Marie Lindquist. Bayesian neural networks with confidence estimations applied to data mining. *Computational Statistics & Data Analysis*, 34(4):473–493, 2000.

[66] Yongchan Kwon, Joong-Ho Won, Beom Joon Kim, and Myunghee Cho Paik. Uncertainty quantification using bayesian neural networks in classification: Application to biomedical image segmentation. *Computational Statistics & Data Analysis*, 142:106816, 2020.

[67] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *arXiv preprint arXiv:1703.04977*, 2017.

[68] Kumar Shridhar, Felix Laumann, and Marcus Liwicki. Uncertainty estimations by softplus normalization in bayesian convolutional neural networks with variational inference. *arXiv preprint arXiv:1806.05978*, 2018.

[69] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

[70] Tim Pearce, Felix Leibfried, and Alexandra Brintrup. Uncertainty in neural networks: Approximately bayesian ensembling. In *International conference on artificial intelligence and statistics*, pages 234–244. PMLR, 2020.

[71] Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*, 2015.

[72] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

[73] Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings*

of the sixth annual conference on Computational learning theory, pages 5–13, 1993.

[74] David JC MacKay. Probable networks and plausible predictions-a review of practical bayesian methods for supervised neural networks. *Network: computation in neural systems*, 6(3):469, 1995.

[75] Radford M Neal. Bayesian training of backpropagation networks by the hybrid monte carlo method. Technical report, Citeseer, 1992.

[76] Alex Graves. Practical variational inference for neural networks. *Advances in neural information processing systems*, 24, 2011.

[77] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[78] Sida Wang and Christopher Manning. Fast dropout training. In *international conference on machine learning*, pages 118–126. PMLR, 2013.

[79] Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. *Advances in neural information processing systems*, 28:2575–2583, 2015.

[80] Christos Louizos and Max Welling. Multiplicative normalizing flows for variational bayesian neural networks. In *International Conference on Machine Learning*, pages 2218–2227. PMLR, 2017.

[81] Chunyuan Li, Andrew Stevens, Changyou Chen, Yunchen Pu, Zhe Gan, and Lawrence Carin. Learning weight uncertainty with stochastic gradient mcmc for shape classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5666–5675, 2016.

[82] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[83] Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

[84] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[85] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013.

[86] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[87] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

[88] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *ICLR 2018, arXiv preprint arXiv:1712.04248*, 2017.

[89] Francesco Croce and Matthias Hein. Sparse and imperceivable adversarial attacks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4724–4732, 2019.

[90] Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*, 2017.

[91] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *ICLR-2017, arXiv preprint arXiv:1611.01236*, 2017.

[92] Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári. Learning with a strong adversary. *arXiv preprint arXiv:1511.03034*, 2015.

[93] Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. *In Advances in Neural Information Processing Systems, Annual Conference on Neural Information Processing Systems*, pages 2263–2273, 2017.

[94] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pages 274–283. PMLR, 2018.

[95] Nilesh Dalvi, Pedro Domingos, Sumit Sanghai, and Deepak Verma. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 99–108, 2004.

[96] Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Analysis of classifiers' robustness to adversarial perturbations. *Machine Learning*, 107(3):481–508, 2018.

[97] Uri Shaham, Yutaro Yamada, and Sahand Negahban. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing*, 307:195–204, 2018.

[98] Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *ICLR 2020*, 2020.

[99] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *ICLR 2018, arXiv preprint arXiv:1705.07204*, 2017.

[100] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

[101] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[102] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[103] Krizhevsky Alex, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional networks. In *volume-1; pages-1097–1105; NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems*.

[104] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *ICLR 2014, arXiv preprint arXiv:1312.6229*, 2013.

[105] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[106] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

[107] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[108] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[109] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[110] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[111] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[112] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3464–3473, 2019.

[113] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. *arXiv preprint arXiv:1906.05909*, 2019.

[114] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3286–3295, 2019.

[115] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10076–10085, 2020.

[116] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. *ICLR 2020, arXiv preprint arXiv:1911.03584*, 2019.

[117] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16519–16529, 2021.

[118] Souvik Kundu and Sairam Sundaresan. Attentionlite: Towards efficient self-attention models for vision. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2225–2229. IEEE, 2021.

[119] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2017.

[120] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019.

[121] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[122] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.

[123] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *International conference on artificial neural networks*, pages 270–279. Springer, 2018.

[124] Jie Lu, Vahid Behbood, Peng Hao, Hua Zuo, Shan Xue, and Guangquan Zhang. Transfer learning using computational intelligence: A survey. *Knowledge-Based Systems*, 80:14–23, 2015.

[125] Ling Shao, Fan Zhu, and Xuelong Li. Transfer learning for visual categorization: A survey. *IEEE transactions on neural networks and learning systems*, 26(5):1019–1034, 2014.

[126] Zhengming Ding and Yun Fu. Robust transfer metric learning for image classification. *IEEE Transactions on Image Processing*, 26(2):660–670, 2016.

[127] Mahbub Hussain, Jordan J Bird, and Diego R Faria. A study on cnn transfer learning for image classification. In *UK Workshop on computational Intelligence*, pages 191–202. Springer, 2018.

[128] Manali Shaha and Meenakshi Pawar. Transfer learning for image classification. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 656–660. IEEE, 2018.

[129] Dongmei Han, Qigang Liu, and Weiguo Fan. A new image classification method using cnn transfer learning and web data augmentation. *Expert Systems with Applications*, 95:43–56, 2018.

[130] Yin Cui, Yang Song, Chen Sun, Andrew Howard, and Serge Belongie. Large scale fine-grained categorization and domain-specific transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4109–4118, 2018.

[131] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[132] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[133] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[134] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[135] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[136] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[137] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.

[138] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[139] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.

[140] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32:103–112, 2019.

[141] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[142] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[143] [Online]. Torchvision.models. Available: https://pytorch.org/vision/stable/models.html.

[144] [Online]. Cifar10 classification using pytorch. Available: https://github.com/huyvnphan/PyTorch_CIFAR10.

[145] [Online]. Efficientnet pytorch. Available: https://github.com/lukemelas/EfficientNet-PyTorch.

[146] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L Yuille, and Quoc V Le. Adversarial examples improve image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 819–828, 2020.

[147] Qiyu Wan and Xin Fu. Fast-bcnn: Massive neuron skipping in bayesian convolutional neural networks. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 229–240. IEEE, 2020.

[148] Ruizhe Cai, Ao Ren, Ning Liu, Caiwen Ding, Luhao Wang, Xuehai Qian, Massoud Pedram, and Yanzhi Wang. Vibnn: Hardware acceleration of bayesian neural networks. *ACM SIGPLAN Notices*, 53(2):476–488, 2018.

[149] Ruizhe Cai, Ao Ren, Luhao Wangy, Massoud Pedramy, and Yanzhi Wang. Hardware acceleration of bayesian neural networks using ram based linear feedback gaussian random number generators. In *2017 IEEE International Conference on Computer Design (ICCD)*, pages 289–296. IEEE, 2017.

[150] Yuki Hirayama, Tetsuya Asai, Masato Motomura, and Shinya Takamaeda-Yamazaki. A resource-efficient weight sampling method for bayesian neural network accelerators. In *2019 Seventh International Symposium on Computing and Networking (CANDAR)*, pages 137–142. IEEE, 2019.

[151] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

List of publications

Peer review journal paper

1. Thi Thu Thao KHONG, Takashi NAKADA, and Yasuhiko NAKASHIMA, "Flexible Bayesian Inference by Weight Transfer for Robust Deep Neural Networks", IEICE Transactions on Information and Systems, Paper, Vol.E104-D, No.11, pp.1981–1991, 2021. (Sections 3 and 4.2.)

Peer review conference paper

1. Thi Thu Thao KHONG, Takashi NAKADA, and Yasuhiko NAKASHIMA, "Bayes without Bayesian Learning for Resisting Adversarial Attacks", 2020 Eighth International Symposium on Computing Networking (CANDAR), Outstanding paper award, pp.221–227, 2020. (Sections 3.1, 3.2.1, 3.2.2, 3.4.2, and 4.1.)