IEICE TRANS. INF. & SYST., VOL.E105–D, NO.7 JULY 2022

PAPER

A Hybrid Bayesian-Convolutional Neural Network for Adversarial Robustness

Thi Thu Thao KHONG[†a)], ***Nonmember***, **Takashi NAKADA**[††], ***Member***, ***and*** **Yasuhiko NAKASHIMA**[†], ***Fellow***

SUMMARY We introduce a hybrid Bayesian-convolutional neural network *(hyBCNN)* for improving the robustness against adversarial attacks and decreasing the computation time in the Bayesian inference phase. Our *hyBCNN* models are built from a part of BNN and CNN. Based on pre-trained CNNs, we only replace convolutional layers and activation function of the initial stage of CNNs with our Bayesian convolutional (BC) and Bayesian activation (BA) layers as a term of transfer learning. We keep the remainder of CNNs unchanged. We adopt the Bayes without Bayesian Learning (BwoBL) algorithm for *hyBCNN* networks to execute Bayesian inference towards adversarial robustness. Our proposal outperforms adversarial training and robust activation function, which are currently the outstanding defense methods of CNNs in the resistance to adversarial attacks such as PGD and C&W. Moreover, the proposed architecture with BwoBL can easily integrate into any pre-trained CNN, especially in scaling networks, e.g., ResNet and EfficientNet, with better performance on large-scale datasets. In particular, under l_∞ norm PGD attack of pixel perturbation $\epsilon = 4/255$ with 100 iterations on ImageNet, our best *hyBCNN* EfficientNet reaches 93.92% top-5 accuracy without additional training.

key words: *Bayesian neural network, convolutional neural network, adversarial robustness, image classification*

1. Introduction

Deep neural networks (DNNs) always exist a certain degree of risk and can be vulnerable to motivated adversaries [1] owing to statistical properties. Adversarial examples are crafted by adding imperceptible perturbations to original images [2] and they make a well-trained neural network misclassify, as seen in Fig. 1. Recent studies have developed several attack methods to generate adversarial images, such as Fast Gradient Sign Method (FGSM) [3], Projected Gradient Descent (PGD) [4], and C&W attack [5]. These are the popular benchmarks of white-box adversarial attacks, which are based on a gradient of the loss function to minimize the perturbation. PGD is an evolution of FGSM. Hence, we focus on white-box and non-targeted settings on both PGD and C&W algorithms in our experiments, which are strong gradient-based iterative attacks currently.

The existence of adversarial instances to image classification exposes a weakness of DNNs. Hence, many effective defenses have been proposed, such as defensive distillation, data augmentation, feature denoising, robust activation functions, and adversarial training. Currently, adversarial training [3], [4] has been the most successful defense of DNNs, which train a model on adversarial images. Moreover, the stochastic components of DNNs have also demonstrated the potential robustness against adversarial attacks [6], [7]. Especially, Bayesian Neural Networks (BNNs) have been indicated as an efficient defense when it is combined with adversarial learning [8], [9]. Nevertheless, Bayesian learning is easy on small datasets, e.g., MNIST, CIFAR-10, and becomes difficult on large-scale datasets like ImageNet. Therefore, Bayesian inference has been considered in the Bayes without Bayesian Learning (BwoBL) algorithm [10], [11], which constructed BNNs based on pre-trained DNNs towards adversarial robustness.

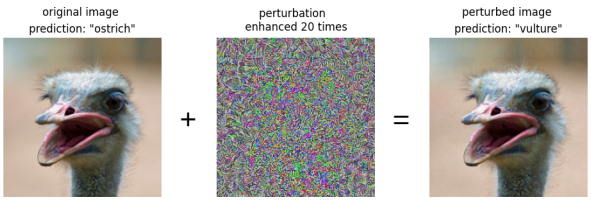


Fig. 1 Adversarial perturbation on ImageNet with ResNet-50 model. Left: the original image is correctly predicted by ResNet-50. Middle: the adversarial perturbation corresponds to the original image. Right: the perturbed image is incorrectly predicted.

BNNs have been built by replacing all convolutional layers of Convolutional Neural Networks (CNNs) with Bayesian convolutional (BC) layers and keeping the rest of CNNs unchanged in [10], [11]. In this paper, we introduce Bayesian activation (BA) layers and combine BC with BA to build BNNs towards the improvement of robustness. Furthermore, changing all convolutional layers and activation functions into BC and BA layers is unnecessary for boosting performance and consumes the inference time. Thus, we propose a hybrid Bayesian-convolutional neural network *(hyBCNN)*. To the best of our knowledge, this proposed architecture is the first hybrid of BNNs and CNNs for adversarial robustness with neither Bayesian learning nor adversarial training. Our contributions can be summarized as follows:

- In addition to BC (Bayesian convolutional), we introduce BA (Bayesian activation) layers and build BNN from BC and BA layers.
- *hyBCNN* is constructed by a part of BNN and CNN. With this construction, we have still generated the stochastic model but do not make the model bigger.

Manuscript received November 11, 2021.
Manuscript revised March 7, 2022.
Manuscript publicized April 11, 2022.

[†]The authors are with Nara Institute of Science and Technology, Ikoma-shi, 630–0192 Japan.
[††]The author is with International Professional University of Technology in Osaka, Osaka-shi, 530–0001 Japan.
a) E-mail: kttthao@hueuni.edu.vn

DOI: 10.1587/transinf.2021EDP7239

- With the application of transfer learning, we build BNN and *hyBCNN* from pre-trained CNNs and utilize learned parameters of CNNs to implement Bayesian inference with the BwoBL algorithm.
- The robustness of the proposed method is evaluated and confirmed with PGD and C&W attacks on ImageNet.

The remainder of this paper is organized as follows. Section 2 introduces related work in BCNNs and adversarial robustness. Section 3 shows the construction of BNNs from BC and BA layers. Section 4 describes the structure of hybrid Bayesian-convolutional neural networks. Section 5 evaluates the accuracy on natural data and the robustness on adversarial examples of the proposed *hyBCNN*. The conclusion is indicated in Sect. 6.

2. Related Work

2.1 Bayesian Convolutional Neural Network

BCNNs are known as stochastic CNNs with the uncertainty estimation on the parameters of the model, which can solve an overconfident decision of CNNs. Generally, a BCNN is built by replacing the weights of convolutional layers in CNN, which are fixed values, with probabilistic distributions, as seen in Fig. 2. In Bayesian methods for DNNs, we aim to estimate the posterior distribution of the weights $\mathbf{w}$ given the observed data D [12]–[14]. However, the exact computation to the posterior over the weights is often intractable. The posterior approximation is thus studied over the past time, in which a variational inference has been proposed by Blundell et al. [15]. In this learning, the variational posterior is assumed to be a Gaussian distribution $q(\mathbf{w}|D) \sim \mathcal{N}(\mu, \sigma^2)$. Monte Carlo sampling has been employed to draw the weight samples from the variational posterior $q(\mathbf{w}|D)$. The parameters can be obtained by sampling a unit Gauss $\xi \sim \mathcal{N}(0, I)$, then shifting by a mean μ and scaling by a standard deviation σ as follows:

$$\mathbf{w} = \mu + \sigma \odot \xi \tag{1}$$

where $\odot$ is point-wise multiplication. We can realize that the variational parameters (μ, σ) are learned during the Bayesian training phase. Besides, the generation of weight samples is generally not easy in the learning process. For these reasons, the learning of BNNs becomes bulky for real datasets and deep neural networks.

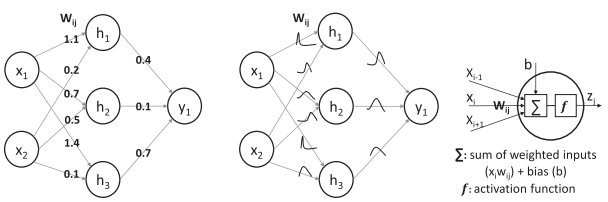


Fig. 2 Comparison between a CNN and a BNN. *Left*: a CNN with specific values of the weights. *Middle*: a BNN with the Gaussian distribution on the weights. *Right*: an inner structure of a neuron with the weights w_{ij} that are single-point values in CNN or Gaussian distribution in BNN.

2.2 Adversarial Attacks

Adversarial examples are usually generated by the gradient-based algorithms to the input to maximize the loss function [3]–[5], [16]–[18]. Particularly, FGSM is a single-step attack algorithm [3], which perturbs the original image x by the direction of the gradient of the loss function $J(\theta, x, y)$, in which y is a label of x, θ is the parameters of a model. A perturbation ϵ is added to each pixel to measure a close (l_∞) distance between x and its adversary x' so that

$$\| x - x' \|_\infty \leq \epsilon \tag{2}$$

It is noted that FGSM is designed to be fast rather than optimal. Therefore, multi-step perturbations have been proposed to produce optimal adversaries. Currently, the PGD attack of Madry et al. [4] is the best multi-step variant of FGSM. The PGD algorithm implements a strong iterative attack to generate adversarial instances, as follows:

$$x^{t+1} = \prod_{\epsilon} \left(x^t + \varepsilon \operatorname{sign} \left(\nabla_x J(\theta, x, y) \right) \right) \tag{3}$$

in which ε is an attack step size, and $\prod_\epsilon$ is a projection to l_∞ norm adversary.

In addition, Carlini and Wagner introduced the C&W approach [5] that dropped the robustness of defensive distillation. Adversarial examples $x' = \frac{1}{2}(\tanh(w) + 1)$ are built by searching for w in l_2 distance to solve the optimization:

$$\min \left\| \frac{1}{2}(\tanh(w) + 1) - x \right\|_2^2 + c \cdot f\left(\frac{1}{2}(\tanh(w) + 1) \right) \tag{4}$$

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

$$f(x') = \max \left(\max_{i, i \neq t} \{ Z(x')_i \} - Z(x')_t, -\kappa \right) \tag{5}$$

The confidence of an adversarial example is adjusted by κ. $Z(\cdot)$ is the output of the network. i and t are the classes of original and adversarial images, respectively. PGD and C&W are the intense attack algorithms, which are used in our experiment.

2.3 Robust Activation Function

When proposing adversarial examples, Szegedy et al. [2] showed blind spots of neural networks that were searched from the properties of non-linear activation functions. We realize that non-linear activation functions are fundamental for DNNs, in which Rectified Linear Units (ReLU) [19] have been widely used and shown the high performance for DNN architectures. Nonetheless, several findings [20], [21] have presented the unbounded, non-smooth, and fixed properties of non-linear activation functions weaken DNNs in

the context of adversarial attacks. In order to improve the robustness of DNNs, a lot of research has focused on the change of activation functions, for example, bounded activation functions [22], [23], data-dependent activation functions [24], quantized activation functions [25], stochastic activation pruning [7]. The randomness on the activation function is mentioned by Dhillon et al. [7], but they have just proved the robustness of DNNs against weak attacks (FGSM) on small datasets (CIFAR-10). Most of these activation functions have just been robust when they are combined with adversarial training. In this paper, we generate the stochastic activation function by applying the Bayesian method to the activation function to resist strong attacks on real datasets like ImageNet without additional training.

2.4 Adversarial Training

Adversarial training has been proposed by Madry et al. [4] that is optimized as below formula:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim D}\left[\max_{\epsilon\in S} J(\theta, x+\epsilon, y)\right] \tag{6}$$

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. This formula shows two computation steps of adversarial training: (1) an inner maximization, which takes adversarial instances, and (2) an outer minimization, which finds model parameters. So as to generate adversarial images, we need a fixed ϵ and several iterations of the gradient computation to achieve optimal adversaries by the PGD algorithm in step (1). These iterations yield a high cost for training. Moreover, if the pixel perturbation ϵ and the number of iterations change, adversarial training needs to be re-trained to improve the robustness. In the outer minimization (2), we must implement many epochs in the training phase to find out the optimal parameters of a model. Both steps make adversarial training consume a high cost of the computation time. Therefore, some studies have focused on training single-step adversaries (FGSM) to reduce the computation time [26], [27]. Nevertheless, FGSM training can make the accuracy of a model suddenly drop when resisting PGD attacks, which is known as catastrophic overfitting because of a lack of various adversarial instances. The incorporation of BNNs and adversarial training has been perceived as a good defense to prevent strong gradient-based attacks. Liu et al. [8] indicated the robustness of Bayesian adversarial training to adversarial attacks on CIFAR-10 and ImageNet-143. However, this training becomes harder on ImageNet due to the disadvantages of Bayesian learning and adversarial training. Consequently, we mainly concentrate on Bayesian inference based on pre-trained DNNs as a term of transfer learning to withstand iterative adversarial attacks.

3. Our Bayesian Neural Networks

3.1 BC and BA Layers

Our BNNs are built on pre-trained CNNs via replacing convolutional and activation layers with BC and BA layers, respectively. Based on Gaussian variational posterior [15] and Gaussian dropout [28], we approximate the variational posterior of parameters with a Gaussian distribution. Particularly, we adopt the variational posterior distribution $q(\mathbf{w}) = \mathcal{N}(\theta, \rho\theta^2)$ with the local reparameterization trick [29]. With this posterior, the variance of $\mathbf{w}$ is tied by its magnitude. A larger weight is then valuable when it is robust to noise. Therefore, instead of Eq. (1), we have a formula of the weight in Bayesian convolutional layers as:

$$\mathbf{w} = \theta + \alpha\theta \odot \xi \tag{7}$$

where ξ is a unit Gauss $\mathcal{N}(0, I)$. We treat θ as single fixed values of learned CNN parameters while $\alpha = \sqrt{\rho}$ is empirically determined by the robustness of our BNN model against the adversarial attack. [11] determined the robustness of BNNs that were constructed by altering all convolutional layers of CNNs with Bayesian convolutional layers.

In this paper, we propose Bayesian activation functions to generate randomness on the output of activation functions. As mentioned in Sect. 2, stochastic activation functions are an efficient defense of DNNs against adversarial attacks. The output of each layer in a neural network can be denoted h_i that consists of a linear transformation W_i followed by a non-linearity ϕ_i as below:

$$h_i = \phi_i(W_i h_{i-1}) \tag{8}$$

It is highlighted that weight matrices and activation functions provide a non-linear mapping from inputs to outputs in a neural network. Our Bayesian activation function is built like the following formula:

$$\Phi = \phi + \beta\phi \odot \xi \tag{9}$$

in which ϕ is the activation function of pre-trained CNNs. With this formula, we yield the uncertain activations Φ from fixed values of the non-linearity ϕ. Similar to Bayesian convolutional layers, the Bayesian activation is approximated by the Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$. We also apply the sampling method followed by the unit Gauss $\xi \sim \mathcal{N}(0, I)$. Accordingly, μ is treated as single-point outputs ϕ of activation layers in pre-trained CNNs. We mainly concentrate on adjusting β to change the standard deviation σ. Intuitively, we draw random samples from the activation map given probabilistic distributions.

In our experiments, we use two activation functions: ReLU and Swish. From Eq. (9), we build the Bayesian ReLU (BReLU) as follows:

$$\begin{aligned} &\text{ReLU: } \phi_R(x_i) = \begin{cases} 0 & \text{if } x_i < 0 \\ x_i & \text{otherwise} \end{cases} \\ &\text{BReLU: } \Phi_{BR}(x) = \phi_R(x) + \beta\phi_R(x) \odot \xi \end{aligned} \tag{10}$$

where $x_i \in x$. We also have the Bayesian Swish (BSwish):

$$\begin{aligned} &\text{Swish: } \phi_S(x) = x \cdot \text{sigmoid}(\gamma x) \\ &\text{BSwish: } \Phi_{BS}(x) = \phi_S(x) + \beta\phi_S(x) \odot \xi \end{aligned} \tag{11}$$

where $\text{sigmoid}(z) = \frac{1}{1+e^{-z}}$, γ is a pre-trained parameter.

From building BC and BA layers based on convolutional and activation layers of CNNs, we use pre-trained CNNs to construct our BCNNs and perform Bayesian inference without additional learning. Much research has proved that the ensemble model outperforms the single model, especially in adversarial robustness [6], [8], [30]. It is known that the ensemble phase is a crux of BNNs [14] because the uncertainty on weights makes BNN equivalent to the ensemble of random models.

3.2 The Search of One Structural Hyperparameter

So as to build our BNNs, we need BC and BA layers with the parameters in Eq. (7) and the non-linearity in Eq. (9). From two equations, we focus on controlling α and β to generate the stochastic models based on pre-trained CNNs. These two parameters empirically depend on pre-trained CNN architectures, they are named the *structural hyperparameter*.

Algorithm 1. The search of structural hyperparameters

Input: small dataset D'
learned parameters of pre-trained CNNs θ
activation of pre-trained CNNs ϕ
Bayesian convolutional layers: $\mathbf{w} = \theta + \alpha\theta \odot \xi, \xi \sim \mathcal{N}(0, I)$
Bayesian activation layers: $\Phi = \phi + \beta\phi \odot \xi$
procedure α and β search
  for α **in** S **do**
    for β **in** S **do**
      for (x, y) **in** D' **do**
        $\hat{x} = attack\_function\ ((\theta, \phi), x, y)$
        for $i = 1, 2, \ldots, N$ **do** # ensemble
          $\hat{y}_i \leftarrow f_\xi((\mathbf{w}, \Phi), \hat{x})$ # N forward passes
        end for
        $\hat{y} = majority\_voting(\hat{y}_i)$ # the most frequent output
      end for
      calculate top-5 accuracy: $top5_{\alpha,\beta}$
    end for
  end for
  optimal $\alpha, \beta \leftarrow \text{argmax}_{\alpha,\beta \in S}\left(top5_{\alpha,\beta}\right)$
end & return optimal α, β

Simply, we treat α and β as one structural hyperparameter for both BC and BA layers, and they are generally called the α *hyperparameter*. The α search algorithm is listed in Algorithm 1, in which $\beta \equiv \alpha$. Let x be an original image and y be a corresponding label in a small dataset D'. We employ l_∞ norm PGD attack to generate an adversarial image $\hat{x}$ and use pre-trained CNNs with learned weights θ and non-linearities ϕ to construct our BNNs with the Gaussian distribution on the weight $\mathbf{w}$ and the activation Φ. The search range S of α is examined in each network. If a pre-trained CNN is denoted $f((\theta, \phi), x)$, our BNN is $f_\xi((\mathbf{w}, \Phi), x)$ in which $\mathbf{w}$ and Φ are sampled by the Monte Carlo sampling method with each probabilistic variable $\xi \sim \mathcal{N}(0, I)$. Therefore, it generates an infinite number of stochastic models by the ensemble inference procedure. As in [10], [11], we also utilize the majority voting scheme to achieve the output of the ensemble. The search process and the ensemble phase consume an amount of computation time, then, we have just executed 10 forward passes ($N = 10$) for the α search.

Algorithm 2. Bayesian inference with BwoBL algorithm

Input: dataset D
learned parameters of pre-trained CNNs θ
activation of pre-trained CNNs ϕ
Initialization: $\xi \sim \mathcal{N}(0, I)$, α, and β
Bayesian convolutional layers: $\mathbf{w} = \theta + \alpha\theta \odot \xi$
Bayesian activation layers: $\Phi = \phi + \beta\phi \odot \xi$
procedure inference
  for (x, y) **in** D **do**
    $\hat{x} = attack\_function\ ((\theta, \phi), x, y)$
    for $i = 1, 2, \ldots$ **do** # n_ensemble
      $\hat{y}_i \leftarrow f_\xi((\mathbf{w}, \Phi), \hat{x})$ # several forward passes
    end for
    $\hat{y} = majority\_voting(\hat{y}_i)$ # the most frequent output
  end for
end procedure

3.3 Bayesian Inference with BwoBL Algorithm

The BwoBL algorithm has been introduced in [10] for implementing Bayesian inference based on pre-trained CNNs. The inference process of our BNNs with the BwoBL algorithm is listed in Algorithm 2. This inference algorithm is executed after searching and fixing structural hyperparameters for each BNN.

From Eq. (7), we can see that a large α leads to a big variance of $\mathbf{w}$ that causes the uncertainty on model parameters. In Eq. (9), β results in probabilistic activation. With this uncertainty, the model is stochastic and efficient to deceive the adversary. In our BNNs, α and β are treated as one structural hyperparameter and are found in Algorithm 1. [11] mentioned that the ensemble was an important property of Bayesian inference by implementing the stochastic model many times. A number of samples for the ensemble also plays an essential role to get better performance. In our experiments, we evaluate 50 samples for ensemble inference with deep networks on ImageNet.

3.4 BNNs with BC and BA Layers

We make a comparison of BNNs with BC and BA layers to determine a robust architecture towards adversarial attacks. For the preliminary assessment, we use ResNet-50 [31], which is trained on natural ImageNet [32] with pixel values of images in [0,255], and employ the cheap PGD attack, which perturbs 4 pixels ($\epsilon = 4/255$) in 10 iterations.

The first BNN is built by replacing all convolutional layers of pre-trained ResNet-50 with BC layers, which has already been proposed in [10], [11], named Full BC. In the second BNN, BA layers (BReLU) are the substitute for all activation layers ReLU [19] of pre-trained ResNet-50,

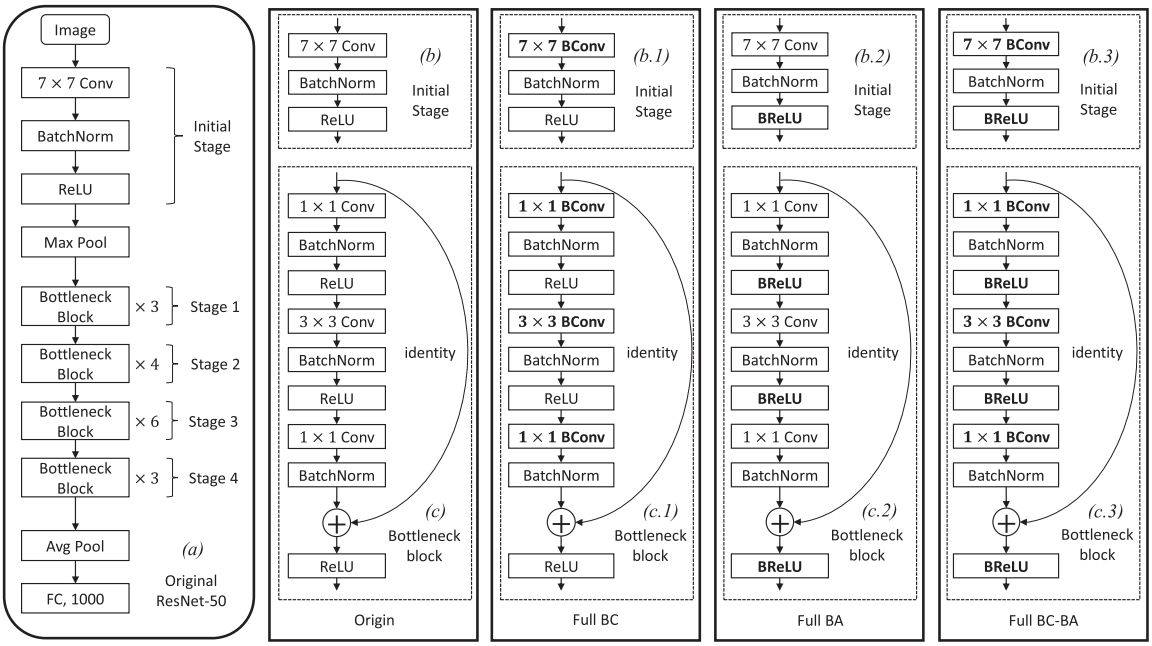


Fig. 3 Our Bayesian ResNet-50 architectures. *(a)* is the original ResNet-50 for ImageNet. *(b)* is the initial stage of the original ResNet-50. *(c)* illustrates the detail of bottleneck blocks in Stages 1, 2, 3, and 4 of the original ResNet-50. *(b.1), (b.2), (b.3)*, and *(c.1), (c.2), (c.3)* are the replacement of *(b)* and *(c)*, respectively. Based on the original network, we build our Bayesian ResNet-50 models as follows: (1) Full BC with the initial stage *(b.1)* and bottleneck blocks *(c.1)*; (2) Full BA with the initial stage *(b.2)* and bottleneck blocks *(c.2)*; (3) Full BC-BA with the initial stage *(b.3)* and bottleneck blocks *(c.3)*. BConv (Bayesian Convolution) and BReLU (Bayesian ReLU) stand for our BC and BA layers. We only replace convolutional and activation layers and keep other layers of the original ResNet-50.

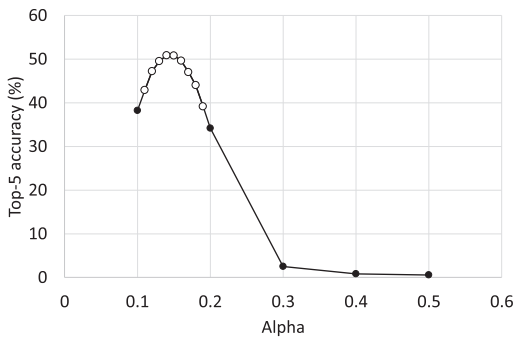


Fig. 4 The change of the α hyperparameter leads to a convex upward of the accuracy. The optimal α corresponds to the peak of this convex upward. Proposed model: Full BC-BA ResNet-50. PGD attack: l_∞ norm, $\epsilon = 4/255$, iteration $it = 10$. Black and white markers denote the step size 0.1 and 0.01 of α.

Table 1 Robustness of three BNNs built on naturally pre-trained ResNet-50: Full BC [11], proposed Full BA, and proposed Full BC-BA resisting the PGD attack: l_∞ norm, $\epsilon = 4/255$, iterations $it = 10$. The ensemble inference of BNNs is executed with 50 samples and the majority voting output. The robustness of models is evaluated by the top-5 accuracy (%). The latency is the average inference time for one image on a single core of Intel®Core™ i9-10920X CPU and a GeForce RTX3090 GPU.

ResNet-50	α	Top-5 (%)	Latency (s)
Pre-trained CNN	0.00	4.86	0.0058
Full BC [11]	0.24	44.92	0.3713
Proposed Full BA	0.18	48.29	0.7433
Proposed Full BC-BA	0.14	52.44	0.8238

which is called Full BA. Our third BNN is the combination of BC and BA layers to replace all convolutional and activation layers of pre-trained ResNet-50, named Full BC-BA. The original ResNet-50 and our Bayesian ResNet-50 with Full BC, Full BA, and Full BC-BA are illustrated in Fig. 3.

In this comparison, we have just utilized one structural hyperparameter α for both BC and BA layers in three BNNs, including Full BC, Full BA, and Full BC-BA. The optimal α is sought as Algorithm 1, which is implemented on the small ImageNet (5000 images of the validation set). The search range S of α depends on each architecture, for example, S = [0.1,0.5] is enough for the α search in Full BC-BA ResNet-50, as seen in Fig. 4. With this S range, we can find out the best robustness of the model against the PGD attack, which is the top-5 accuracy = 50.9% corresponding to $\alpha = 0.14$. Likewise, applying Algorithm 1 for Full BC ResNet-50 and Full BA ResNet-50, we get $\alpha = 0.24$ and 0.18, respectively.

We fix the optimal α for each architecture before the evaluation of the inference phase. The ensemble inference for three BNNs is followed by Algorithm 2. We generate the PGD attack on ImageNet with 50000 images of the validation set and execute the ensemble inference with 50 forward passes. Table 1 proves that applying the Bayesian method to activation layers further improves the robustness of naturally pre-trained CNN, which increases by 3.37% top-5 accuracy compared to Full BC. Moreover, the combination of Full

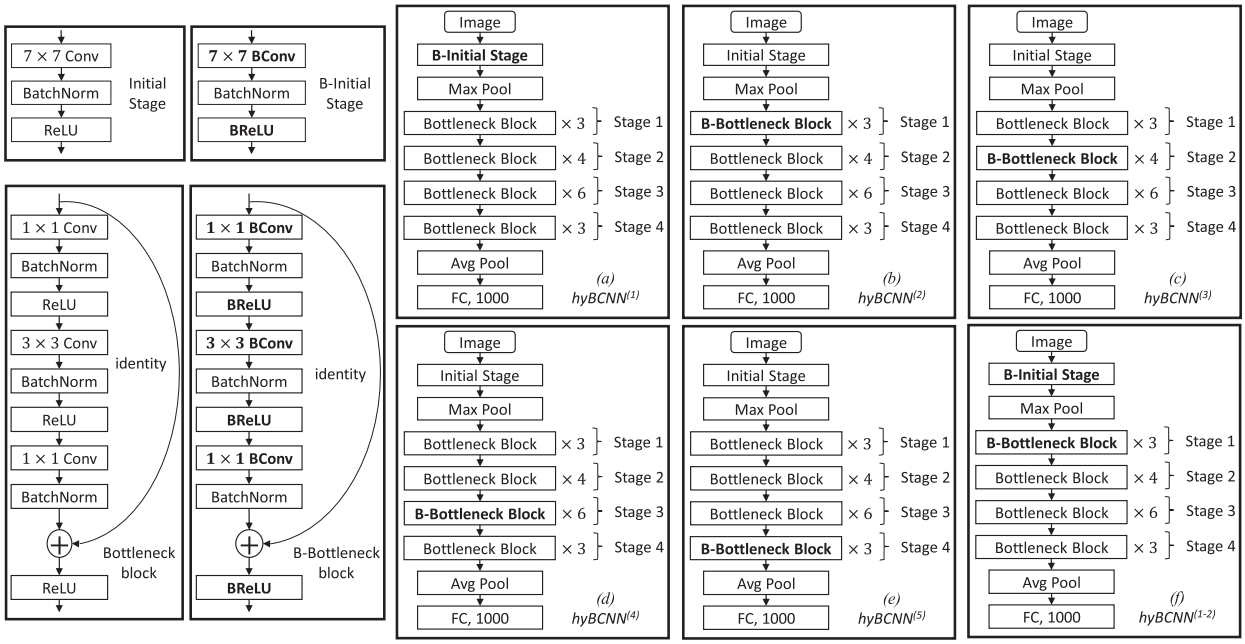


Fig. 5 Structure of hybrid ResNet-50 (*hyBCNN*). From the original network, hybrid models are built by sequentially applying Bayesian convolution (BConv) and Bayesian ReLU (BReLU) layers to the stages: Initial stage, Stages 1, 2, 3, and 4.

BC-BA networks sharply boosts the robustness of naturally pre-trained CNNs, which achieves 52.44% accuracy against the PGD attack. Accordingly, our BNN with Bayesian convolutional and activation layers via the BwoBL algorithm is an efficient defense with neither Bayesian learning nor adversarial training. Nevertheless, the computation time is a weakness of Bayesian inference by the ensemble phase. We can compare the latency of models in Table 1. The latency of BNNs is much larger than that of CNN. Among BNNs, the latency of Full BC-BA is the biggest. It means the enhancement of the robustness is difficult to reduce the inference time of Bayesian models. For that reason, we propose a hybrid Bayesian-convolutional neural network, which is the hybrid of BNN and CNN to decrease the number of Bayesian layers in the architecture.

4. Our Hybrid Bayesian-Convolutional Neural Networks

So as to build BNNs towards adversarial robustness and reduce the computation time of the ensemble inference, we introduce a hybrid Bayesian-convolutional neural network (*hyBCNN*) that consists of a part of BNN and CNN based on pre-trained CNNs.

4.1 Structure of *hyBCNN*

To search the powerful hybrid architecture against the PGD attack (l_∞ norm, $\epsilon = 4/255$, 10 iterations) on ImageNet, we implement our experiment with ResNet-50, which is trained on natural ImageNet. Based on the original ResNet-50 in Fig. 3 (a), we construct hybrid models by sequentially applying both BC and BA layers to the stages: the initial stage, Stages 1, 2, 3, and 4, as shown in Fig. 5. We have hybrid BCNNs as follows:

- *hyBCNN*$^{(1)}$: Convolutional and activation layers of the initial stage are replaced by BConv and BReLU layers as Fig. 5 (a). The rest of original ResNet-50 is kept.
- *hyBCNN*$^{(2)}$: Convolution and activation layers of bottleneck blocks in Stage 1 are replaced by BConv and BReLU layers as Fig. 5 (b). We keep the remainder of original ResNet-50 unchanged.
- With the same substitution as *hyBCNN*$^{(2)}$, we achieve *hyBCNN*$^{(3)}$, *hyBCNN*$^{(4)}$, and *hyBCNN*$^{(5)}$ via the replacement of BC and BA layers of bottleneck blocks in Stages 2, 3, and 4, respectively, as seen in Fig. 5 (c, d, e). In each hybrid model, only one stage is altered to generate a part of BNN, the rest parts are CNN.

With these hybrid structures, we seek the best position of BC and BA layers that is the first, the middle, or the last in existing CNN architectures. In each hybrid BCNN, the part of BNN is built from BC and BA layers with two hyperparameters α and β that need to be found. Nonetheless, to simplify the search algorithm, we behave α and β as one hyperparameter and employ the same search as Sect. 3.2. The optimal α of each hybrid network is presented in Table 2. Based on the top-5 accuracy of *hyBCNN*$^{(1),(2),(3),(4),(5)}$, it is confirmed that hybrid BCNNs with the first part of BNN achieve better robustness resisting adversarial attacks. Then, could the robustness be further improved if we apply BC and BA layers to both the initial stage and Stage 1? Accordingly, we construct *hyBCNN*$^{(1,2)}$, as shown in Fig. 5 (f), and execute its experiments like the above hybrid architec-

Table 2 Robustness of hybrid BCNNs built on naturally pre-trained ResNet-50 against the PGD attack: l_∞ norm, $\epsilon = 4/255$, iterations $it = 10$. The ensemble inference of each hybrid BCNN is executed with 50 samples. The robustness of models is evaluated by the top-5 accuracy (%). The latency is the average inference time for one image on a single core of Intel®Core™ i9-10920X CPU and a GeForce RTX3090 GPU.

ResNet-50	α	Top-5 (%)	Latency (s)
$hyBCNN^{(1)}$	0.24	57.75	0.2779
$hyBCNN^{(2)}$	0.27	43.74	0.3996
$hyBCNN^{(3)}$	0.34	21.39	0.4028
$hyBCNN^{(4)}$	0.35	9.84	0.4253
$hyBCNN^{(5)}$	0.40	5.16	0.3361
$hyBCNN^{(1,2)}$	0.18	57.51	0.4254

tures. As a result in Table 2, the robustness of $hyBCNN^{(1,2)}$ is not enhanced and it takes a long inference time. Among hybrid BCNNs in Table 2, we can see the application of BC and BA layers to the initial stage of original ResNet-50 not only bring the best robustness but also lessen the ensemble inference time.

Furthermore, we compare Table 1 to Table 2, it is convinced that the hybrid BCNN can improve both the robustness and the inference time. Particularly, compared to Full BC-BA, the top-5 accuracy of $hyBCNN^{(1)}$ rises by 5.31%, and $hyBCNN^{(1)}$ runs 2.96× faster. Between Tables 1 and 2, $hyBCNN^{(1)}$ is the best architecture, we then utilize it to continue examining its efficiency in advanced CNN models under strong PGD and C&W attacks. In the next sections, our hybrid model is shortly called *hyBCNN*, which implies $hyBCNN^{(1)}$.

4.2 *hyBCNN* with Two Structural Hyperparameters

We find out the best *hyBCNN* with treating α and β of BC and BA layers as one hyperparameter in Sect. 4.1. Then, what happens if α and β are two independent hyperparameters? It means we must seek α and β separately and their search algorithm becomes complicated with the loop.

Empirically, we take a search space $S = [0.01, 0.9]$ for α and β with a step size 0.05, and implement Algorithm 1 with *hyBCNN* built on naturally pre-trained ResNet-50. We also generate the PGD attack (l_∞ norm, $\epsilon = 4/255$, iteration = 10) on random 5000 images of the validation set of ImageNet and execute the ensemble inference over 10 samples. Figure 6 illustrates a part of the search space of α and β, which guarantees the optimal search algorithm with convex upwards of the accuracy. Hence, this search algorithm brings the optimal pair of α and β, which is the highest peak of convex upwards. For example, with $\alpha = 0.2$ and $\beta = 0.25$ in Fig. 6, the accuracy of hybrid ResNet-50 reaches the peak. After fixing the optimal α and β, we carry out Bayesian inference of hybrid ResNet-50 on the validation set of ImageNet followed Algorithm 2. We execute the ensemble phase with 50 forward passes and assess the top-5 accuracy of *hyBCNN* that is built on pre-trained ResNet-50 on natural ImageNet. We make a comparison of hybrid ResNet-50 between using α and a pair (α, β) to build BC and BA layers. The fact that the employment of two hyperparameters is

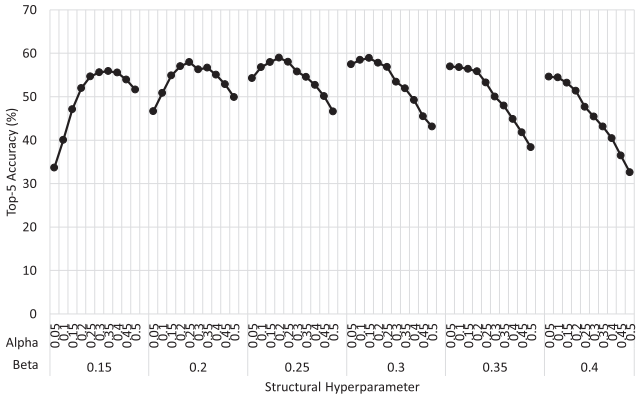


Fig. 6 A part of the search space of α and β, which results in convex upwards of the accuracy. An optimal pair of α and β corresponds to the highest peak of convex upwards. Proposed model: *hyBCNN* on pre-trained ResNet-50. PGD attack: l_∞ norm, $\epsilon = 4/255$, iteration $it = 10$.

Table 3 Comparison of *hyBCNN* between using one and two structural hyperparameters. *hyBCNN* is built on naturally pre-trained ResNet-50 and evaluated by the top-5 accuracy on natural (nat.) and adversarial (adv.) ImageNet.

one hyperparameter			two hyperparameters			
α	Top-5 (%)		α	β	Top-5 (%)	
	nat.	adv.			nat.	adv.
0.24	77.63	57.75	0.20	0.25	80.21	58.54

more complex and consumes the computation time for the search loop. However, Table 3 indicates the treatment of the pair (α, β), which is separate for BC and BA layers of *hyBCNN*, can improve both the robustness on adversarial images and the accuracy on natural images. Compared to hybrid ResNet-50 (α), the accuracy of hybrid ResNet-50 (α, β) increases by 2.58% and the robustness against the PGD attack (l_∞ norm, $\epsilon = 4/255$, iteration = 10) raises by 0.79%. Therefore, building the *hyBCNN* model with the application of BC (α) and BA (β) layers to the initial stage of CNN architectures is promising in the enhancement of robustness and accuracy.

In order to verify the performance of our *hyBCNN* model, in the next section, we implement *hyBCNN* on deeper ResNets and EfficientNet family, which are the state-of-the-art CNNs trained on natural ImageNet.

5. Accuracy and Robustness of *hyBCNN*

5.1 *hyBCNN* Models

We build up *hyBCNN* models from the advanced CNN architectures currently, i.e., ResNets and EfficientNets, which achieve the advanced accuracy on ImageNet [33]. This is the real dataset for the task of classifying 1000 classes with large-scale images.

Baseline CNN models used in our experiments are ResNet-18, ResNet-50, ResNet-152 [31], ResNeXt-101-32×8d [34], and EfficientNet-B0~B7 [35]. ResNet models demonstrate deeper and wider networks can further improve

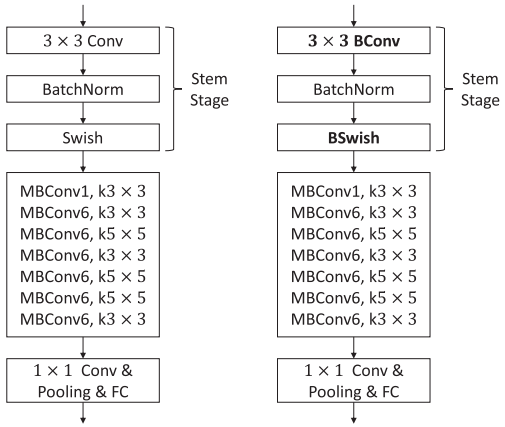


Fig. 7 Architecture of EfficientNet-B0 (left) and our hybrid EfficientNet-B0 (right). We replace convolutional (Conv) and activation (Swish) layers of the stem stage of original EfficientNet by Bayesian convolutional (BConv) and Bayesian activation (BSwish) layers in our hybrid EfficientNet. MBConv denotes mobile inverted bottleneck blocks.

the performance of CNN in image recognition. Moreover, the image resolution is also vital for boosting the accuracy of CNNs. A compound scaling method, which balanced three elements of networks, i.e., depth/width/resolution, has been proposed by Tan et al. [35]. They investigated the state-of-the-art model for real datasets, which is a family of EfficientNet. We use pre-trained ResNets [32] and EfficientNets [36] on natural ImageNet to build our *hyBCNN* models and execute Bayesian inference with the BwoBL algorithm. Additionally, the BwoBL algorithm can apply to pre-trained networks on cheap adversarial images to boost the robustness without adding adversarial training, as determined in [11]. Therefore, we also construct *hyBCNN* models on adversarial pre-trained EfficientNets, which is AdvProp scheme proposed by Xie et al. [37] and we call them EfficientNet-Adv.

The initial stage of ResNet-18, ResNet-152, and ResNeXt-101-32×8d are the same as ResNet-50 (as seen in Fig. 3 (b)), they are only different in the remaining stages. Our hybrid ResNet-18, ResNet-152, and ResNeXt-101-32×8d only apply BC (α) and BA (β) layers to the initial stage, they are thus like hybrid ResNet-50. The remainder stages of original ResNet-18, ResNet-152 and ResNeXt-101-32×8d are maintained. In EfficientNets, the stem stage is initial, as shown in Fig. 7. We also build hybrid EfficientNets with the substitution of BC and BA layers to this stem stage and the rest stages are kept. Swish activation [38], which works better than ReLU on deeper networks, is used in EfficientNet. Hence, we form Bayesian Swish (BSwish) activation to substitute for Swish activation. The stem stage is the same from EfficientNet-B0 to B7 and illustrated in Fig. 7 (left), we thus employ hybrid EfficientNet in Fig. 7 (right) to construct our hybrid EfficientNet-B0~B7.

So as to validate our methodology, we compare the robustness of *hyBCNN* with adversarial training, robust activation function, and BCNN with BwoBL. So far, adversarial training is still an outstanding defense towards strong adversarial attacks. In which, Fast adversarial training [27] is a fast training method on FGSM adversaries with ResNet-50, but can resist PGD attacks on ImageNet. We call it ResNet-50-Fast in our experiments. The second trend to adversarial robustness is robust activation functions. In particular, SPLASH activation [21], which does not need adversarial training but can withstand powerful attacks like C&W on ImageNet, is compared to ReLU and Swish. Tavakoli et al. trained ResNet-18 with ReLU, Swish, and SPLASH on natural ImageNet and assessed the robustness to C&W attacks. Lastly, we compare the robustness of BCNN [11] and *hyBCNN* via the BwoBL algorithm. Our experiments are run on four GeForce GTX1080Ti and two GeForce RTX3090.

Table 4 *hyBCNN* networks built on pre-trained CNNs and structural hyperparameters (α, β) of each *hyBCNN*. Eff-Net-B0~B7 denote EfficientNet-B0~B7, which are trained on natural ImageNet. Eff-Net-Adv-B0~B7 are EfficientNet-Adv-B0~B7, which are trained on both natural and cheap adversarial ImageNet.

***hyBCNN* on naturally pre-trained CNNs**	α	β	***hyBCNN* Eff-Net-Adv**	α	β
ResNet-18	0.10	0.45			
ResNet-50	0.20	0.25			
ResNet-152	0.10	0.25			
ResNeXt-101-32×8d	0.25	0.15			
Eff-Net-B0	0.40	0.15	B0	0.25	0.05
Eff-Net-B1	0.45	0.15	B1	0.30	0.10
Eff-Net-B2	0.45	0.15	B2	0.35	0.10
Eff-Net-B3	0.65	0.15	B3	0.40	0.10
Eff-Net-B4	0.70	0.20	B4	0.50	0.15
Eff-Net-B5	0.70	0.30	B5	0.40	0.20
Eff-Net-B6	0.65	0.35	B6	0.45	0.35
Eff-Net-B7	0.50	0.45	B7	0.40	0.20

5.2 Adversarial Attacks and Structural Hyperparameters

To verify the robustness of our *hyBCNN* networks, we employ two strong adversarial attacks that are PGD and C&W. Following Eq. (3), we generate PGD attacks: l_∞ norm, pixel perturbation $\epsilon = 4/255$ as Fast training [27], the attack iteration $it = \{10, 100\}$. Based on Eq. (4) of C&W, we set attack parameters as follows: l_2 norm, a binary search step of 7 for 1000 steps as SPLASH activation [21]. Both PGD and C&W are white-box and non-targeted attacks.

We perform the search of two structural hyperparameters followed Sect. 4.2, which are α for BC layers and β for BA layers in all our *hyBCNN* models. We test each *hyBCNN* with Algorithm 1 and observe the best pairs of (α, β), as indicated in Table 4. We fix the pairs of (α, β) for each *hyBCNN* before the main Bayesian inference.

Our Bayesian inference of all *hyBCNN* networks is executed as Algorithm 2. We assess 50 forward passes and the majority voting output for the ensemble phase in each *hyBCNN*.

5.3 Accuracy on Natural ImageNet

We assess the top-5 accuracy of all our *hyBCNN* models on natural ImageNet with 50000 images of the validation

Table 5 The accuracy on natural images and the robustness under PGD attacks: l_∞ norm, pixel perturbation $\epsilon = 4/255$, iterations $it = \{10, 100\}$. We evaluate the top-5 accuracy (%) and compare our *hyBCNN* to naturally pre-trained CNNs, Full BC networks [11], Fast adversarial training [27].

	Natural images			**PGD attack of** $it = 10$			**PGD attack of** $it = 100$		
	pre-trained CNNs	**Full BC**	**proposed *hyBCNN***	**pre-trained CNNs**	**Full BC**	**proposed *hyBCNN***	**pre-trained CNNs**	**Full BC**	**proposed *hyBCNN***
ResNet-18	89.31	71.63	71.59	1.44	24.40	46.72	0.29	24.79	49.41
ResNet-50	92.89	81.39	80.21	4.78	44.92	58.54	2.50	46.90	61.45
ResNet-152	94.05	84.13	84.33	6.36	54.47	63.82	3.59	58.15	68.25
ResNeXt-101-32×8d	94.56	87.84	87.37	13.95	65.57	71.00	10.75	67.66	74.72
EfficientNet-B0	91.71	76.37	75.61	0.58	27.03	50.57	0.09	32.26	55.27
EfficientNet-B1	91.49	75.53	80.87	0.25	25.84	56.45	0.06	32.14	64.54
EfficientNet-B2	94.40	80.11	86.96	0.57	45.36	72.78	0.20	52.34	77.81
EfficientNet-B3	95.27	86.73	90.16	0.79	53.00	77.26	0.32	61.83	82.69
EfficientNet-B4	96.16	89.87	92.89	0.61	65.60	80.62	0.45	73.48	87.06
EfficientNet-B5	96.77	89.68	93.66	0.66	71.44	84.07	0.14	78.40	89.09
EfficientNet-B6	96.90	91.16	93.86	0.28	68.68	84.16	0.16	78.50	89.86
EfficientNet-B7	96.97	91.65	93.18	0.21	71.19	85.18	0.02	81.62	89.49
EfficientNet-Adv-B0	89.46	80.15	80.96	12.78	46.98	59.14	8.04	49.86	60.85
EfficientNet-Adv-B1	93.29	87.75	87.92	18.28	64.06	75.29	10.27	65.46	76.90
EfficientNet-Adv-B2	94.44	91.17	91.39	18.31	73.78	81.41	10.16	76.20	82.46
EfficientNet-Adv-B3	95.04	91.30	93.31	29.98	81.19	84.89	17.48	81.54	85.90
EfficientNet-Adv-B4	95.85	93.13	93.69	30.33	85.54	88.54	15.63	86.48	89.17
EfficientNet-Adv-B5	96.80	94.79	95.32	32.75	89.65	91.83	13.37	90.40	92.15
EfficientNet-Adv-B6	97.04	95.34	96.06	31.83	90.92	93.11	10.05	91.69	93.74
EfficientNet-Adv-B7	97.14	95.52	96.30	29.66	91.58	**93.33**	7.67	92.66	**93.92**
Fast adversarial training on pixel perturbation $\epsilon = 4/255$									
ResNet-50-Fast [27]	77.22			56.85			56.09		

set. Many studies of adversarial training indicated that gaining robustness of the model would be losing accuracy on natural examples. Likewise, the BwoBL algorithm, which is performed in Full BC neural networks [11] and our *hyBCNN*, is not an exception. This method mainly focuses on improving the robustness of BNNs to the adversaries rather than natural images. Nevertheless, compared to Fast adversarial training [27], our *hyBCNN* can enhance both the accuracy and the robustness. A comparison of the top-5 accuracy between naturally pre-trained CNNs, Full BC [11], Fast adversarial training [27], and proposed *hyBCNN* is shown in Table 5. Compared to adversarial training on the same pixel perturbation $\epsilon = 4/255$, i.e., ResNet-50-Fast, the accuracy of our *hyBCNN* ResNet-50 increases by 2.99%. Moreover, our *hyBCNN* is built on naturally pre-trained CNNs, but its uncertainty is towards the adversarial robustness. Thus, the accuracy of *hyBCNN* ResNet-50 significantly decrease from 92.89% to 80.21%. The accuracy of *hyBCNN* ResNet-50 also declines by 1.18% compared to Full BC while its robustness is significantly boosted. Otherwise, we realize that deeper and wider networks yield competitive accuracies on natural images, our *hyBCNN* is then applied to the advanced CNNs. In particular, the top-5 prediction of our *hyBCNN* EfficientNet-Adv-B7, which is 96.30%, is the advanced accuracy on natural ImageNet among defense methods currently.

5.4 Robustness to PGD Attacks

We evaluate the robustness of our proposed *hyBCNN* models under l_∞ norm PGD attack of pixel perturbation $\epsilon = 4/255$ and iterations $it = \{10, 100\}$. It is known that increasing the iteration of the PGD attack generates optimal adversaries. In order to validate our method, we compare the proposed *hyBCNN* to Fast adversarial training [27], and Full BC neural networks [11]. Both Full BC networks and our *hyBCNN* apply the BwoBL algorithm to naturally pre-trained CNNs against gradient-based attacks without adversarial training or Bayesian learning.

With the same application of the BwoBL algorithm [10], our *hyBCNN* is better than Full BC networks [11], as shown in Table 5. The top-5 accuracy of our *hyBCNN* ResNet-50 increases by 13.62% and 14.55% compared to Full BC ResNet-50 resisting PGD attacks of the pixel perturbation $\epsilon = 4/255$ and the iterations $it = \{10, 100\}$, respectively. Fast adversarial training achieves the 56.85% prediction with ResNet-50-Fast under the PGD attack of $it = 10$ and decreases by 0.76% when increasing the attack iteration $it = 100$. In contrast, the accuracy of proposed *hyBCNN* ResNet-50 is 58.54% under the PGD attack of $it = 10$. Our *hyBCNN* ResNet-50 becomes robust towards the PGD attack of $it = 100$ with the 61.45% prediction. To the best of our knowledge, 61.45% is the best accuracy of naturally pre-trained ResNet-50 under the strong PGD attack (l_∞ norm, $\epsilon = 4/255, it = 100$) among defense methods currently.

Table 5 also proves that scaling up CNNs by the depth/width/resolution can considerably boost the adversarial robustness. Large-scale models are often difficult to adversarial training, but our proposal is easy to improve the robustness of these networks without learning. For instance, under the PGD attack of $\epsilon = 4/255$ and $it = 100$, naturally pre-trained ResNet-152 and ResNeXt-101-32×8d only achieve 3.59% and 10.75% accuracies, but our *hyBCNN*

ResNet-152 and ResNeXt-101-32×8d with the BwoBL algorithm can reach 68.25% and 74.72% predictions. Compared to Full BC networks, our *hyBCNN* ResNet-152 and ResNeXt-101-32×8d rise by 8% on average.

Pre-trained EfficientNets on natural images are almost fooled by PGD attacks, whose top-5 accuracy is less than 1%. However, our *hyBCNN* EfficientNets with the BwoBL algorithm, which are based on naturally pre-trained EfficientNets without adversarial training, increase by 79.3% on average compared to pre-trained networks and 18.2% on average compared to Full BC networks, under PGD attack of 100 iterations. Furthermore, EfficientNet-Adv is the model that is trained on cheap PGD adversaries, e.g., $\epsilon = 4/255, it = 5$, and fine-tuned on natural images to enhance image recognition. Owing to the advantage of pre-training on both natural and adversarial examples, our *hyBCNN* EfficientNet-Adv strongly boosts the robustness of pre-trained EfficientNet-Adv. From Table 5, we can see the accuracy of pre-trained EfficientNet-Adv only obtain a maximum of 32.75% and 17.48% under PGD attacks of $it = 10$ and $it = 100$, respectively. Meanwhile, our *hyBCNN* EfficientNet-Adv-B7 reach a maximum of 93.33% and 93.92% under these PGD attacks. Accordingly, our *hyBCNN* with the BwoBL algorithm is conveniently applied to pre-trained CNNs on natural images and weak adversaries to sharply improve the robustness of pre-trained models against strong attacks without additional adversarial training.

5.5 Robustness to C&W Attacks

It is known that the convergence of C&W is slower than that of PGD due to the iteration of the optimal constant search as Eq. (4). It hence expends considerable time to generate C&W attacks. For that reason, we assess the performance of all networks on random 1000 images, which are correctly classified from the validation set of ImageNet. We generate adversarial examples over three runs and evaluate the number of success attacks in the form of *mean* ± *standard deviation*. In comparison with no adversarial training and an influence of activation function, our *hyBCNN* outperforms the robust SPLASH activation [21].

Even though structural hyperparameters (α, β) of BC and BA layers in our *hyBCNN* are determined based on the PGD attack, the proposed *hyBCNN* is still robust under strong C&W attack, as seen in Table 6. Tavakoli et al. prove that SPLASH is more robust than ReLU, Swish in ResNet-18 against the C&W attack when all of them is trained on natural images. Nonetheless, our *hyBCNN* ResNet-18, which is built with Bayesian ReLU activation and based on naturally pre-trained ResNet-18+ReLU, significantly boosts the robustness of pre-trained CNNs without additional training. Compared to ResNet-18-SPLASH against the C&W attack, the number of success attacks of our *hyBCNN* ResNet-18 is reduced by about one second. Notice that both ResNet-18-SPLASH and our *hyBCNN* ResNet-18 use a naturally pre-trained model towards adversarial robustness.

Table 6 Robustness of pre-trained CNNs, Full BC networks [11], SPLASH activation [21], and our *hyBCNN* networks to l_2 norm C&W attack, which is assessed by *mean* ± *standard deviation* of the number of success attacks on correctly classified 1000 images of pre-trained CNNs. Each model is attacked 3 times.

	Pre-trained CNNs	**Full BC**	**proposed *hyBCNN***
ResNet-18	1000 ± 0	501 ± 9.18	419 ± 14.45
ResNet-50	1000 ± 0	378 ± 14.27	363 ± 7.94
ResNet-152	1000 ± 0	370 ± 6.19	325 ± 18.63
ResNeXt-101-32×8d	1000 ± 0	282 ± 6.19	251 ± 9.27
EfficientNet-B0	1000 ± 0	590 ± 8.06	474 ± 9.09
EfficientNet-B1	1000 ± 0	581 ± 11.52	342 ± 14.98
EfficientNet-B2	1000 ± 0	470 ± 15.97	261 ± 10.41
EfficientNet-B3	1000 ± 0	375 ± 2.94	217 ± 11.33
EfficientNet-B4	1000 ± 0	266 ± 10.41	161 ± 9.00
EfficientNet-B5	1000 ± 0	241 ± 4.12	137 ± 11.56
EfficientNet-B6	1000 ± 0	246 ± 3.42	132 ± 5.57
EfficientNet-B7	1000 ± 0	227 ± 7.79	136 ± 3.74
EfficientNet-Adv-B0	1000 ± 0	776 ± 6.95	563 ± 9.04
EfficientNet-Adv-B1	1000 ± 0	596 ± 10.42	375 ± 8.96
EfficientNet-Adv-B2	1000 ± 0	473 ± 14.14	287 ± 14.99
EfficientNet-Adv-B3	1000 ± 0	378 ± 21.06	266 ± 6.03
EfficientNet-Adv-B4	1000 ± 0	273 ± 18.57	185 ± 8.74
EfficientNet-Adv-B5	1000 ± 0	174 ± 5.00	126 ± 6.86
EfficientNet-Adv-B6	1000 ± 0	155 ± 10.42	99 ± 3.27
EfficientNet-Adv-B7	1000 ± 0	129 ± 4.97	**95 ± 2.94**
SPLASH activation function trained on natural images			
ResNet-18-SPLASH [21]	929 ± 1.8		

The proposed *hyBCNN* networks, which are built on scaling models such as ResNet-50, ResNet-152, ResNeXt-101-32×8d, EfficientNets, continue exposing their high robustness against the C&W attack. The number of success attacks of these *hyBCNN* models is very low while their pre-trained networks are almost attacked. Particularly, pre-trained EfficientNet-Adv-B7, which is trained on natural images and cheap PGD adversaries, is completely fooled by the C&W attack but our *hyBCNN* EfficientNet-Adv-B7 resists that attack strongly as seen in Table 6.

Tables 5 and 6 verify the best robustness of our *hyBCNN* models with the BwoBL algorithm resisting both PGD and C&W attacks even when we construct them based on pre-trained CNNs without additional training.

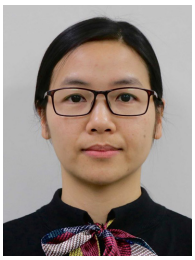
6. Conclusion

In this paper, we discover a hybrid Bayesian-convolutional neural network, which is built on various pre-trained CNNs and applied the BwoBL algorithm against adversarial attacks. Our method builds *hyBCNN* models on the advanced pre-trained CNNs via replacing convolutional layers and activation function of the initial stage of pre-trained CNNs with our BC and BA layers. We focus on controlling a pair of structural hyperparameters of BC and BA layers to generate the stochastic models and execute Bayesian inference towards adversarial robustness. We hereby declare the proposed *hyBCNN* networks with the BwoBL algorithm as an effective resistance to a variety of adversaries without costing adversarial Bayesian learning. In particular, under l_∞ norm PGD attack of pixel perturbation $\epsilon = 4/255$

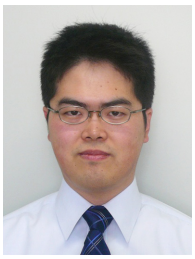
with 100 iterations on ImageNet, the top-5 accuracy of our best *hyBCNN* reaches 93.92%, which is the most robustness among defense methods. Our *hyBCNN* models not only resist strong PGD attacks but also are robust towards intense C&W attacks and promising for other gradient-based attacks in the future. Moreover, the proposed algorithm is based on learned parameters, it is thus expected to be a good defense for other datasets as long as pre-trained DNNs exist.

References

[1] C. Chio and D. Freeman, Machine learning and security: Protecting systems with data and algorithms, O'Reilly Media, Inc., 2018.
[2] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2nd International Conference on Learning Representations - ICLR 2014, arXiv:1312.6199, pp.1–10, 2013.
[3] I.J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 3rd International Conference on Learning Representations - ICLR 2015, arXiv:1412.6572, pp.1–11, 2014.
[4] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 6th International Conference on Learning Representations - ICLR 2018, arXiv:1706.06083, pp.1–28, 2017.
[5] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," 2017 ieee symposium on security and privacy (sp), pp.39–57, IEEE, 2017.
[6] X. Liu, M. Cheng, H. Zhang, and C.-J. Hsieh, "Towards robust neural networks via random self-ensemble," Proceedings of the European Conference on Computer Vision (ECCV), pp.369–385, 2018.
[7] G.S. Dhillon, K. Azizzadenesheli, Z.C. Lipton, J. Bernstein, J. Kossaifi, A. Khanna, and A. Anandkumar, "Stochastic activation pruning for robust adversarial defense," ICLR 2018, arXiv:1803.01442, pp.1–13, 2018.
[8] X. Liu, Y. Li, C. Wu, and C.J. Hsieh, "Adv-bnn: Improved adversarial defense through robust bayesian neural network," ICLR 2019, arXiv:1810.01279, pp.1–13, 2018.
[9] N. Ye and Z. Zhu, "Bayesian adversarial learning," Proceedings of the 32nd international conference on neural information processing systems, pp.6892–6901, 2018.
[10] T.T.T. Khong, T. Nakada, and Y. Nakashima, "Bayes without bayesian learning for resisting adversarial attacks," 2020 Eighth International Symposium on Computing and Networking (CANDAR), pp.221–227, IEEE, 2020.
[11] T.T.T. Khong, T. Nakada, and Y. Nakashima, "Flexible bayesian inference by weight transfer for robust deep neural networks," IEICE Transactions on Information and Systems, vol.E104-D, no.11, pp.1981–1991, 2021.
[12] D.J. MacKay, "Bayesian methods for neural networks: Theory and application," Neural Networks Summer School, University of Cambridge, pp.1–43, 1995.
[13] R.M. Neal, Bayesian learning for neural networks, Springer Science & Business Media, 2012.
[14] D. Barber and C.M. Bishop, "Ensemble learning in bayesian neural networks," Nato ASI Series F Computer and Systems Sciences, vol.168, pp.215–238, 1998.
[15] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," International Conference on Machine Learning, pp.1613–1622, PMLR, 2015.
[16] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," ICLR Workshop 2017, arXiv 1607.02533, pp.1–14, 2017.
[17] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z.B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," 2016 IEEE European symposium on security and privacy (EuroS&P), pp.372–387, IEEE, 2016.
[18] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," Proceedings of the IEEE conference on computer vision and pattern recognition, pp.2574–2582, 2016.
[19] V. Nair and G.E. Hinton, "Rectified linear units improve restricted boltzmann machines," Icml, pp.1–8, 2010.
[20] C. Xie, M. Tan, B. Gong, A. Yuille, and Q.V. Le, "Smooth adversarial training," arXiv preprint arXiv:2006.14536, pp.1–11, 2020.
[21] M. Tavakoli, F. Agostinelli, and P. Baldi, "Splash: Learnable activation functions for improving accuracy and adversarial robustness," Neural Networks, vol.140, pp.1–12, 2021.
[22] H. Zhang, T.W. Weng, P.Y. Chen, C.J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," 32nd Conference on Neural Information Processing Systems (NIPS 2018), arXiv:1811.00866, pp.1–17, 2018.
[23] A. Rozsa and T.E. Boult, "Improved adversarial robustness by reducing open space risk via tent activations," arXiv preprint arXiv:1908.02435, pp.1–15, 2019.
[24] B. Wang, A.T. Lin, W. Zhu, P. Yin, A.L. Bertozzi, and S.J. Osher, "Adversarial defense via data dependent activation function and total variation minimization," arXiv preprint arXiv:1809.08516, pp.1–17, 2018.
[25] A.S. Rakin, J. Yi, B. Gong, and D. Fan, "Defend deep neural networks against adversarial examples via fixed and dynamic quantized activation functions," arXiv preprint arXiv:1807.06714, pp.1–15, 2018.
[26] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L.S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!," 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), arXiv:1904.12843, pp.1–12, 2019.
[27] E. Wong, L. Rice, and J.Z. Kolter, "Fast is better than free: Revisiting adversarial training," ICLR 2020, arXiv:2001.03994, pp.1–17, 2020.
[28] S. Wang and C. Manning, "Fast dropout training," international conference on machine learning, PMLR, pp.118–126, 2013.
[29] D.P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," Advances in neural information processing systems, vol.28, pp.2575–2583, 2015.
[30] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," ICLR 2018, arXiv:1705.07204, pp.1–22, 2017.
[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," Proceedings of the IEEE conference on computer vision and pattern recognition, pp.770–778, 2016.
[32] [Online], "Torchvision.models." Available: https://pytorch.org/vision/stable/models.html
[33] J. Deng, W. Dong, R. Socher, L.J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," 2009 IEEE conference on computer vision and pattern recognition, pp.248–255, Ieee, 2009.
[34] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," Proceedings of the IEEE conference on computer vision and pattern recognition, pp.1492–1500, 2017.
[35] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," International Conference on Machine Learning, pp.6105–6114, PMLR, 2019.
[36] [Online], "Efficientnet." Available: https://github.com/lukemelas/EfficientNet-PyTorch
[37] C. Xie, M. Tan, B. Gong, J. Wang, A.L. Yuille, and Q.V. Le, "Adversarial examples improve image recognition," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp.819–828, 2020.
[38] P. Ramachandran, B. Zoph, and Q.V. Le, "Searching for activation functions," arXiv preprint arXiv:1710.05941, pp.1–13, 2017.



Thi Thu Thao Khong received the M.E. degree from VNU University of Engineering and Technology, Ha Noi, Vietnam in 2014. She has been a Ph.D. student in Division of Information Science, Nara Institute of Science and Technology, Japan since 2019. Her research interests include machine learning, signal processing, modeling and simulation.



Takashi Nakada received his M.E. and Ph.D. degrees from Toyohashi University of Technology in 2004 and 2007, respectively. He was an Associate Professor at the Nara Institute of Science and Technology from 2016 to 2021. He has been an Associate Professor at the International Professional University of Technology in Osaka since 2021. His research interests include Normally-Off Computing, system architecture and related simulation technologies. He is a member of IEEE, ACM, and IPSJ.

Yasuhiko Nakashima received the B.E., M.E., and Ph.D. degrees in Computer Engineering from Kyoto University in 1986, 1988, and 1998, respectively. He was a computer architect in the Computer and System Architecture Department, FUJITSU Limited from 1988 to 1999. From 1999 to 2005, he was an Associate Professor in the Graduate School of Economics, Kyoto University. Since 2006, he has been a Professor in the Graduate School of Information Science, Nara Institute of Science and Technology. His research interests include processor architecture, emulation, CMOS circuit design, and evolutionary computation. He is a member of IEEE CS, ACM, and IPSJ.