Adversarial Examples Detection With Bayesian Neural Network

Yao Li, Tongyi Tang, Cho-Jui Hsieh, and Thomas C. M. Lee, *Senior Member, IEEE*

***Abstract*—In this paper, we propose a new framework to detect adversarial examples motivated by the observations that random components can improve the smoothness of predictors and make it easier to simulate the output distribution of a deep neural network. With these observations, we propose a novel Bayesian adversarial example detector, short for BATER, to improve the performance of adversarial example detection. Specifically, we study the distributional difference of hidden layer output between natural and adversarial examples, and propose to use the randomness of the Bayesian neural network to simulate hidden layer output distribution and leverage the distribution dispersion to detect adversarial examples. The advantage of a Bayesian neural network is that the output is stochastic while a deep neural network without random components does not have such characteristics. Empirical results on several benchmark datasets against popular attacks show that the proposed BATER outperforms the state-of-the-art detectors in adversarial example detection.**

***Index Terms*—Adversarial example, Bayesian neural network, deep neural network, detection.**

I. INTRODUCTION

DESPITE achieving tremendous successes, Deep Neural Networks (DNNs) have been shown to be vulnerable against adversarial attacks [1], [2], [3], [4], [5], [6]. By adding imperceptible perturbations to the original inputs, the attackers can craft adversarial examples to fool a trained classifier. Adversarial examples are indistinguishable from the original inputs to humans but are mis-classified by the classifier. The wide application of machine learning models causes concerns about the reliability and safety of machine learning systems in security-sensitive areas, such as self-driving, financial systems, and healthcare.

Manuscript received 27 November 2023; revised 16 January 2024; accepted 10 February 2024. This work was supported in part by the National Science Foundation under Grant CCF-1934568, Grant IIS-2048280, Grant IIS-2008173, Grant DMS-2113605, Grant DMS-2210388, Grant DMS-2152289, and Grant DMS-2134107, and in part by Cisco Faculty Award. *(Corresponding author: Yao Li.)*

Yao Li is with the Statistics and Operations Research Department, University of North Carolina at Chapel Hill, Chapel Hill, NC 27599 USA (e-mail: yaoli@email.unc.edu).

Tongyi Tang and Thomas C. M. Lee are with the Statistics Department, University of California, Davis, Davis, CA 95616 USA (e-mail: tyitang@ucdavis.edu; tcmlee@ucdavis.edu).

Cho-Jui Hsieh is with the Computer Science Department, University of California, Los Angeles, Los Angeles, CA 90095 USA (e-mail: chohsieh@cs.ucla.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TETCI.2024.3372383, provided by the authors.

Recommended for acceptance by Prof. H. Huang.

Digital Object Identifier 10.1109/TETCI.2024.3372383

There has been extensive research on improving the robustness of deep neural networks against adversarial examples [7], [8], [9], [10]. In [11], the authors showed that many defense methods [12], [13], [14], [15], [16] can be circumvented by strong attacks except Madry's adversarial training [17], in which adversarial examples are generated during training and added back to the training set. Since then, adversarial training-based algorithms have become state-of-the-art methods for defending against adversarial examples. However, despite being able to improve robustness under strong attacks, adversarial training-based algorithms are time-consuming due to the cost of generating adversarial examples on-the-fly. Improving the robustness of deep neural networks remains an open question.

Due to the difficulty of defense, recent work has turned to attempting to detect adversarial examples as an alternative solution. The main assumption made by the detectors is that adversarial samples come from a distribution that is different from the natural data distribution, that is, adversarial samples do not lie on the data manifold, and DNNs perform correctly only near the manifold of the training data [18]. Many works have been done to study the characteristics of adversarial examples and leverage the characteristics to detect adversarial examples instead of trying to classify them correctly [13], [19], [20], [21], [22], [23], [24], [25].

Despite many algorithms that have been proposed for adversarial detection, most of them are deterministic, which means they can only use the information from one single forward pass to detect adversarial examples. This makes it easier for an attacker to break those models, especially when the attacker knows the neural network architecture and weights. In this paper, we propose a novel algorithm to detect adversarial examples based on randomized neural networks. Intuitively, incorporating randomness in neural networks can improve the smoothness of predictors, thus enabling stronger robustness guarantees (see randomized-based defense methods in [16], [26], [27]). Further, instead of observing only one hidden feature for each layer, a randomized network can lead to a distribution of hidden features, making it easier to detect an out-of-manifold example.

a) Contribution and Novelty: We propose a detection method based on Bayesian Neural Network (BNN), leveraging the randomness of BNN to improve detection performance (see the framework in Fig. 1). BNN and some other random components have been used to improve robust classification accuracy [16], [26], [27], [28], [29], [30], [31], but they were not used to improve adversarial detection performance. The proposed method BATER is motivated by the following observations: 1) the hidden

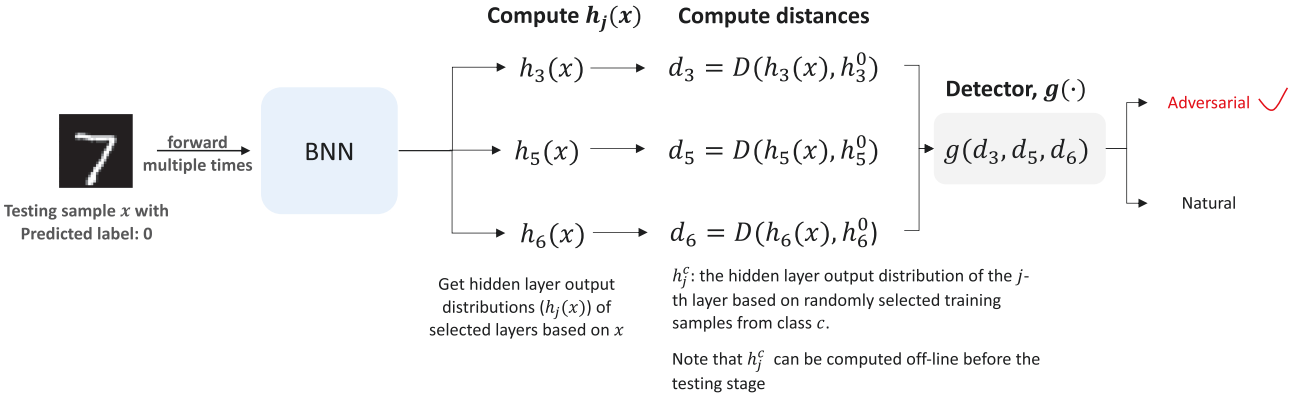


Fig. 1. Detection framework of BATER. An example is given in this diagram to show how BATER works. An adversarial image ($\boldsymbol{x}$) with handwritten digit 7 is mis-classified as 0 by the classifier (BNN). To check if the input is adversarial or not, the input image is fed into the BNN multiple times to get hidden layer output distributions ($h_j(\boldsymbol{x})$) of selected layers (layers 3, 5, and 6 in this example). Details of $h_j(\boldsymbol{x})$ computation and layer selection are given in Section III. Then, distances (d_j) between hidden layer output distributions ($h_j(\boldsymbol{x})$) and hidden layer output distributions based on training samples of predicted class (h_j^c) are computed. In this example, it is h_j^0 because the model predicts the input as class 0. Finally, the distances are fed into the detector to do binary classification: adversarial vs. natural. Details of distance computation and detector training can be found in Section III.

layer output generated from adversarial examples demonstrates different characteristics from that generated from natural data and this phenomenon is more obvious in BNN than in deterministic deep neural networks; 2) randomness of BNN makes it easier to simulate the characteristics of hidden layer output. Training BNN is not very time-consuming as it only doubles the number of parameters of the deep neural network with the same structure [32]. However, BNN can achieve comparable classification accuracy and improve the smoothness of the classifier. A theoretical analysis is provided to show the advantage of BNN over DNN in adversarial detection.

In numerical experiments, our method achieves better performance in detecting adversarial examples generated from popular attack methods on MNIST, CIFAR10 and ImageNet-Sub among state-of-the-art detection methods. Ablation experiments show that BNN performs better than deterministic neural networks under the same detection scheme. Besides, the proposed method is also tested against attacks with different parameters, transfer attacks, and an adaptive attack. In all the tested scenarios, the proposed method can achieve reasonable performance.

b) Notation: In this paper, all the vectors are represented as bold symbols. The input to the classifier is represented by $\boldsymbol{x}$ and the label associated with the input is represented by y. Thus, one observation is a pair $(\boldsymbol{x}, y)$. The classifier is denoted as $f(\cdot)$ and $f(\boldsymbol{x})$ represents the output vector of the classifier. $f(\boldsymbol{x})_i$ is the score of predicting $\boldsymbol{x}$ with label i. The prediction of the classifier is denoted as $c(\boldsymbol{x}) = \text{argmax}_i f(\boldsymbol{x})_i$; that is, the predicted label is the one with the highest prediction score. We use the ℓ_∞ and ℓ_2 distortion metrics to measure similarity and report the ℓ_∞ distance in the normalized [0,1] space (e.g., a distortion of 0.031 corresponds to $8/256$), and the ℓ_2 distance as the total root-mean-square distortion normalized by the total number of pixels [6].

II. RELATED WORK

a) Adversarial attack: Multiple attack methods have been introduced for crafting adversarial examples to attack deep neural networks [11], [33], [34], [35], [36], [37], [38], [39], [40]. Depending on the information available to the adversary, attack methods can be divided into white-box attacks and black-box attacks. Under the white-box setting, the adversary is allowed to analytically compute the model's gradients/parameters, and has full access to the model architecture. Most white-box attacks generate adversarial examples based on the gradient of the loss function with respect to the input [17], [35], [41], [42], [42], [43]. Among them FGSM [1], C & W [35] and PGD [17] attacks have been widely used to test the robustness of machine learning models. In reality, the detailed model information, such as the gradient, may not be available to the attackers [6]. Some attack methods are more agnostic and only rely on the predicted labels or scores [44], [45], [46], [47], [48]. In [44], the authors proposed a method to estimate the gradient based on the score information and craft adversarial examples with the estimated gradient. Some other works [45], [46], [47], [48], [49], [50] introduced methods that also only rely on the final decision of the model.

b) Adversarial defense: To defend against adversarial examples, many studies have been done to improve the robustness of deep neural networks, including adversarial training [17], [51], [52], [53], [54], generative models [14], [55], [56], [57], [58], verifiable defense [59], [60] and other techniques [25], [61], [62], [63], [64], [65]. The authors of [11] showed that many defense methods [12], [13], [14], [15], [16] could be circumvented by strong attacks except Madry's adversarial training [17]. Since then, adversarial training-based algorithms have become state-of-the-art methods in defending against adversarial examples. However, adversarial training is computationally expensive and time-consuming due to the cost of generating adversarial examples on-the-fly, thus adversarial defense is still an open problem to solve.

c) Adversarial detection: Another popular line of research focuses on screening out adversarial examples [25], [66], [67], [68], [69]. A straightforward way towards adversarial example detection is to build a simple binary classifier separating the

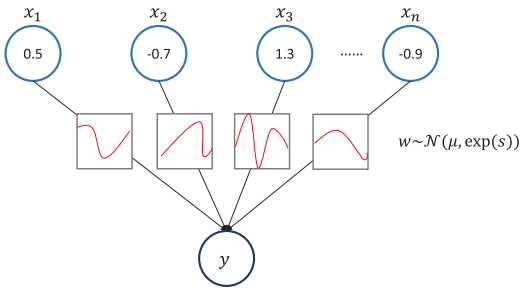


Fig. 2. Illustration of Bayesian Neural Network. All weights in a BNN are represented by probability distributions over possible values, rather than having a single fixed value. The red curves in the graph represent distributions. We view a BNN as a probabilistic model: given an input $\boldsymbol{x}$, a BNN assigns a probability to each possible output y, using the set of parameters $\boldsymbol{w}$ sampled from the learned distributions.

adversarial apart from the clean data based on the characteristics of adversarial examples [19], [23], [66], [70], [71], [72], [73]. In [25], a detection method is implemented based on the consensus of the classifications of the augmented examples, which are generated based on an individually implemented intensity exchange on the red, green, and blue components of the input image. In [19], the author proposed to perform kernel density estimation on the training data in the feature space of the last hidden layer to help detect adversarial examples (KD). The authors of [13] observed that the Local Intrinsic Dimensions (LID) of hidden-layer outputs differ between the original inputs and adversarial examples, and leveraged these findings to detect adversarial examples. In [23], an adversarial detection method based on Mahalanobis distance (MAHA) is proposed. Class conditional Gaussian distributions are first fitted based on the hidden layer output features of the deep neural network, then confidence scores are calculated to compute Mahalanobis distance. In [24], the author studied the feature attributions of adversarial examples and proposed a detection method (ML-LOO) based on feature attribution scores. The author of [74] showed that adversarial examples exist in cone-like regions in very specific directions from their corresponding natural inputs and proposed a new test statistic to detect adversarial examples with the findings (ODD). Recently, a joint statistical test pooling information from multiple layers is proposed in [75] to detect adversarial examples (JTLA). We show that BATER performs comparable or superior to these detection methods across multiple benchmark datasets.

Recently, there has been a shift in focus towards detecting adversarial examples that are generated using black-box methods [76], which are recognized as more realistic threats. Despite being well explored in the vision domain, adversarial example detection started to get attention in the field of natural language processing (NLP) recently [77], [78], [79], [80]. In addition to the domain of NLP, adversarial detection has been extended to the physical world, aiming to identify adversarial examples in real-world scenarios [81].

d) Bayesian neural network: The idea of BNN is illustrated in Fig. 2. In [32], the author introduced an efficient algorithm to learn the parameters of BNN. Given the observable random variables $(\boldsymbol{x}, y)$, BNN aims to estimate the distributions of hidden variables $\boldsymbol{w}$, instead of estimating the maximum likelihood value $\boldsymbol{w}_{\text{MLE}}$ for the weights. Since, in the Bayesian perspective, each parameter is now a random variable measuring the uncertainty of the estimation, the model can potentially extract more information to support a better prediction (in terms of precision, robustness, etc.).

Given the input $\boldsymbol{x}$ and label y, a BNN aims to estimate the posterior over the weights $p(\boldsymbol{w}|\boldsymbol{x}, y)$ given the prior $p(\boldsymbol{w})$. The true posterior can be approximated by a parametric distribution $q_{\boldsymbol{\theta}}(\boldsymbol{w})$, where the unknown parameter $\boldsymbol{\theta}$ is estimated by minimizing the KL divergence

$$\mathsf{KL}\left(q_{\boldsymbol{\theta}}(\boldsymbol{w}) \parallel p(\boldsymbol{w}|\boldsymbol{x}, y)\right) \tag{1}$$

over $\boldsymbol{\theta}$. For simplicity, $q_{\boldsymbol{\theta}}$ is often assumed to be a fully factorized Gaussian distribution:

$$q_{\boldsymbol{\theta}}(\boldsymbol{w}) = \prod_{i=1}^{d} q_{\boldsymbol{\theta}_i}(\boldsymbol{w}_i), \text{ and } q_{\boldsymbol{\theta}_i}(\boldsymbol{w}_i) = \mathcal{N}(\boldsymbol{w}_i; \boldsymbol{\mu}_i, \exp(\boldsymbol{s}_i)^2), \tag{2}$$

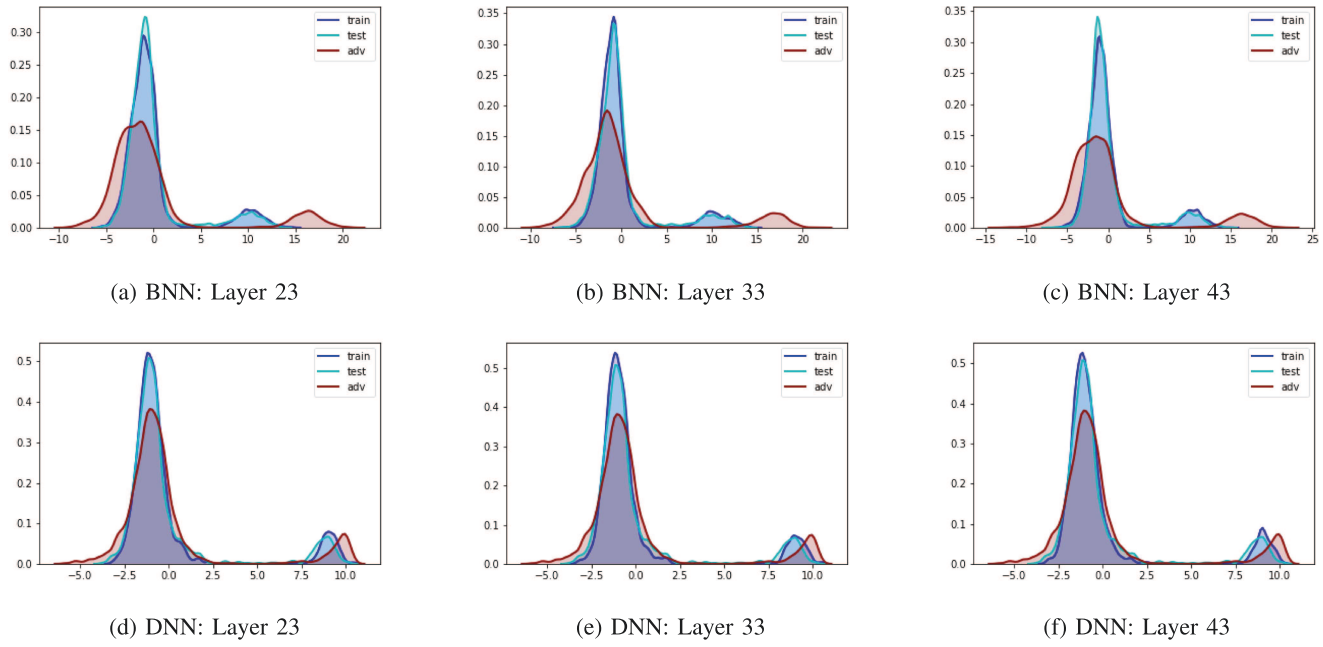
where $\boldsymbol{\mu}$ and $\boldsymbol{s}$ are parameters of the Gaussian distributions of weight. The objective function for training BNN is reformulated from expression (1) and is shown in expression (3), which is a sum of a data-dependent part and a regularization part:

$$\begin{aligned}\underset{\boldsymbol{\mu},\boldsymbol{s}}{\arg\max} \Bigg\{ \sum_{(\boldsymbol{x}_i, y_i) \in \boldsymbol{D}} & \mathbb{E}_{\boldsymbol{w} \sim q_{\boldsymbol{\mu},\boldsymbol{s}}} \log p(y_i|\boldsymbol{x}_i, \boldsymbol{w}) \\ & - \mathsf{KL}\left(q_{\boldsymbol{\mu},\boldsymbol{s}}(\boldsymbol{w}) \| p(\boldsymbol{w})\right) \Bigg\}, \end{aligned} \tag{3}$$

where $\boldsymbol{D}$ represents the data distribution. In the first term of objective (3), the probability of y_i given $\boldsymbol{x}_i$ and weights is the output of the model. This part represents the classification loss. The second term of objective (3) is trying to minimize the divergence between the prior and the parametric distribution, which can be viewed as regularization [32]. The author of [30] showed that the posterior average of the gradients of BNN makes it more robust than DNN against gradient-based adversarial attacks. Though the idea of using BNN to improve robustness against adversarial examples is not new [28], [29], the previous works did not leverage BNN to help detect adversarial examples. In [28], [29], BNN was combined with adversarial training [17] to improve robust classification accuracy.

III. PROPOSED METHOD

We first discuss the motivation behind the proposed method: 1) the distributions of the hidden layer neurons of a deep neural network can be different when based on adversarial examples versus natural images; 2) this dispersion is more obvious in BNN than DNN; 3) it is easier to simulate hidden layer output distribution with random components. Then, we introduce the specific metric used to measure this distributional difference and extend the detection method to multiple layers to make it more resistant to adversarial attacks.



(a) BNN: Layer 23 (b) BNN: Layer 33 (c) BNN: Layer 43

(d) DNN: Layer 23 (e) DNN: Layer 33 (f) DNN: Layer 43

Fig. 3. Hidden Layer output Distributions (HLDs) of VGG16 and BNN (VGG16 based architecture) based on images from automobile class of CIFAR10. Legend explanation: train represents HLDs of training samples from automobile class; test denotes HLDs of testing samples from automobile class; adv shows HLDs of adversarial examples predicted as automobiles. The adversarial examples are generated by PGD [17]. The three plots in the first row show hidden layer distributions of a BNN, and the plots in the second row show the distributions of a DNN with the same base architecture. For both DNN and BNN, there are distributional differences between natural (train and test) and adversarial (adv) hidden outputs, but the differences are larger for BNN.

A. *Motivation: Distributional Difference of Natural and Adversarial Hidden Layer Outputs*

Given input $\boldsymbol{x}$ and a classifier $f(\cdot)$, the prediction of the classifier is denoted as $c(\boldsymbol{x}) = \text{argmax}_i f(\boldsymbol{x})_i$; that is, the predicted label is the one with the highest prediction score. The adversary aims to perturb the original input to change the predicted label:

$$c(\boldsymbol{x}) \neq \underset{i}{\text{argmax}}\, f(\boldsymbol{x} + \boldsymbol{\delta})_i,$$

where $\boldsymbol{\delta}$ denotes the perturbation added to the original input. The attacker aims to find a small $\boldsymbol{\delta}$ (usually lies within a small ℓ_p norm ball) to successfully change the prediction of the model. Thus, given the same predicted label, there could be a distributional difference in hidden layer outputs between adversarial examples and natural data. For example, adversarial examples mis-classified as airplanes could have hidden layer output distributions different from those of natural airplane images. Here, we define a hidden layer output distribution in DNN as the empirical distribution of all the neuron values of that layer, which means all output values of that layer will be used to draw an one-dimensional histogram to simulate the hidden layer distribution. For BNN, a similar approach is used to estimate the hidden layer output distribution. Meanwhile, in BNN, the same input will be forwarded multiple times as the weights of BNN are stochastic to get a better estimation of the output distribution.

In the exploratory analysis, we compare the hidden layer output distributions of DNN and BNN based on both natural and adversarial examples, and find some interesting patterns that are later used in the proposed method. Some examples of hidden layer output distribution comparisons are shown in Fig. 3. The figure shows the hidden layer output distributions of layer 23, layer 33 and layer 43 in DNN and BNN. Blue and cyan (train and test) curves represent distributions of the natural automobile images in CIFAR10. Red curves represent the distributions of adversarial examples mis-classified as automobiles. The adversarial examples are generated by PGD [17] with ℓ_∞ norm. The architecture of the DNN is VGG16 [82] and the architecture of the BNN is also VGG16 [82] except that the weights in BNN follow Gaussian distributions. Both networks are trained on CIFAR10 train set.

In Fig. 3, we can see that for all three hidden layers, there are differences between distributions based on natural and adversarial images. In BNN, the hidden layer output distributions of the natural images (train or test) are clearly different from those of adversarial examples (adv), while the pattern is not that obvious in DNN. Even though hidden layer output distributions of only three layers are shown here, similar patterns are observed in some other layers in BNN. This phenomenon is not a special case with PGD adversarial examples on CIFAR10. Such characteristics are also found in adversarial examples generated by different attack methods on other datasets.

a) Why BNN not DNN?: Differences between distributions based on natural and adversarial examples can be observed in both DNN and BNN. However, the distributional difference is more obvious in BNN than in neural networks without random components (see Fig. 3). Therefore, more information can be extracted from BNN than from deterministic neural networks. Furthermore, random components of BNN make it easier to simulate the hidden layer output distributions. Our experimental results also show that the proposed detection method works

better with BNN than with deterministic neural networks on multiple datasets (see Section IV-B for more details).

Fig. 3 empirically shows the intuition behind the proposed framework. The following theoretical analysis shows that randomness can help enlarge the distributional differences between natural and adversarial hidden layer outputs.

Proposition 1: Let $f(\boldsymbol{x}, \boldsymbol{w})$ be a model with $\boldsymbol{x} \sim \boldsymbol{D_x}$ and $\boldsymbol{w} \sim \boldsymbol{D_w}$, where $\boldsymbol{D_w}$ is any distribution that satisfies $\boldsymbol{w}$ is symmetric about $\boldsymbol{w}_0 = \mathbb{E}[\boldsymbol{w}]$, such as $\mathcal{N}(\boldsymbol{w}_0, \boldsymbol{I})$. If $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{w})$ can be approximated by the first order Taylor expansion at $\boldsymbol{w}_0$, we have

$$\mathcal{D}(f(\boldsymbol{x}+\boldsymbol{\delta}, \boldsymbol{w}), f(\boldsymbol{x}, \boldsymbol{w})) \geq \mathcal{D}(f(\boldsymbol{x}+\boldsymbol{\delta}, \boldsymbol{w}_0), f(\boldsymbol{x}, \boldsymbol{w}_0)), \tag{4}$$

where $\boldsymbol{\delta}$ represents adversarial perturbation and $\mathcal{D}$ represents a translation-invariant distance measuring distribution dispersion (See proof of the inequality in Appendix F).

The inequality shows that randomness involved in parameters will enlarge the distributional differences between natural and adversarial outputs. Therefore, leveraging the hidden layer output distributional differences of BNN to detect adversarial examples is a sensible choice.

B. Detect Adversarial Examples by Distribution Distance

We propose to measure the dispersion between hidden layer output distributions of adversarial examples and natural inputs and use this characteristic to detect adversarial examples. In particular, given an input $\boldsymbol{x}$ and its predicted label c, we measure the distribution distance between the hidden layer output distribution of $\boldsymbol{x}$ and the corresponding hidden layer output distribution of training samples from class c:

$$d_j(\boldsymbol{x}) = \mathcal{D}\left(h_j(\boldsymbol{x}), h_j(\{\boldsymbol{x}_i^c\}_{i=1}^{n_c})\right), \tag{5}$$

where $h_j(\boldsymbol{x})$ represents the hidden layer output distribution of the j-th layer based on testing sample $\boldsymbol{x}$, $h_j(\{\boldsymbol{x}_i^c\}_{i=1}^{n_c})$ represents the hidden layer output distribution of the j-th layer based on training samples from class c, n_c is the number of training samples in class c, and $\mathcal{D}$ can be arbitrary divergence. For simplicity, $h_j(\{\boldsymbol{x}_i^c\}_{i=1}^{n_c})$ is replaced by h_j^c in the rest part of the paper. Besides, n_c does not have to be the total number of training samples in class c. In our experiments, n_c is just a small amount sampled from the training samples of class c. As for the measure of divergence, we estimate the divergence with 1-Wasserstein distance in our experiments. However, other divergence measures can also be used, such as the Kullback–Leibler divergence.

The hidden layer output distribution is estimated by a one-dimensional empirical distribution of all the output values of that layer. The hidden layer output distribution (h_j^c) estimated with training samples of each class can be easily simulated since there are multiple samples in each class. However, at the testing stage, only one testing sample ($\boldsymbol{x}$) is available for the simulation of $h_j(\boldsymbol{x})$. For a deep neural network without random components, the hidden layer output is deterministic, thus the simulation result depends on a single forward pass. For BNN, the hidden layer output is stochastic, thus we can simulate the distribution with multiple passes.

Algorithm 1: BATER.

Input: Input $\boldsymbol{x}$, pre-trained BNN $f(\cdot)$, pre-trained binary classifier $g(\cdot)$, number of passes to simulate hidden layer output distribution B, indices of hidden layers selected for detection $\mathcal{S}$ and divergence $\mathcal{D}$.

Output: Adversarial ($z = 1$) or Natural ($z = 0$).

$c = \text{argmax}_i f(\boldsymbol{x})_i$ ▷ *get the predicted label* c

for $j \in \mathcal{S}$ **do**

  Feed $\boldsymbol{x}$ into $f(\cdot)$ B times to simulate $h_j(\boldsymbol{x})$

  $d_j = \mathcal{D}(h_j(\boldsymbol{x}), h_j^c)$ ▷ h_j^c *is the* j*-th layer output distribution of class* c

$z = g(d_1, d_2, \ldots, d_k)$ ▷ $z = 1$ *indicating adversarial example and* $z = 0$ *indicating natural input*

To pool the information from different levels, the dispersion is measured at multiple hidden layers to generate a set of dispersion scores $\{d_j | j \in \mathcal{S}\}$, where $\mathcal{S}$ is the index set of selected hidden layers (see details of layer selection in Section III-C). It is expected that natural inputs will have small dispersion scores while adversarial examples will have relatively large dispersion scores. A binary classifier is trained on the dispersion scores to detect adversarial examples. In the paper, we fit a binomial logistic regression model to do the binary classification. An overview of the detection framework at testing time is shown in Fig. 1. Details of the method are included in Algorithm 1.

C. Implementation Details

a) Layer Selection: For adversarial examples generated with different attacks on different datasets, the pattern of distributional differences can be different. For example, adversarial examples generated by PGD on CIFAR10 show larger distributional dispersion in deeper layers (layers closer to the final layer). However, such characteristic does not appear in adversarial examples generated by C & W on CIFAR10. Instead, the distributional dispersion is more obvious in some front layers (layers closer to the input layer). Therefore, we develop an automated hidden layer selection scheme to find the layers with large deviations between natural data and adversarial examples. Cross-validation is performed to do layer selection by fitting a binary classifier (logistic regression) with a single layer's dispersion score. Layers with top-ranked performance measured by AUC (Area Under the receiver operating characteristic Curve) scores are selected, and information from those layers is pooled for ultimate detection (See details of selected layers in Appendix C).

b) Distance Calculation: To measure the dispersion between hidden layer output distributions of natural and adversarial samples, we treat the output of a hidden layer as a realization of a one-dimensional random variable. The dispersion between two distributions is estimated by 1-Wasserstein distance between their empirical distributions. In BNN, the empirical distribution of a testing sample can be simulated by multiple forward passes. Whereas, in DNN, a single forward pass is

done to simulate the empirical distribution as the output is deterministic. Training samples from the same class can be used to simulate empirical hidden layer output distributions of natural data of that class. Given a testing sample and its predicted label, calculating the dispersion score with all training samples in the predicted class is expensive, so we sample some natural images in the predicted class as representatives to speed up the process.

c) Dimension Reduction: To further improve computational efficiency, we apply dimension reduction on the hidden layer output. PCA (Principal Component Analysis) is applied to the hidden layer output of training samples to do dimension reduction before the testing stage. At the testing stage, hidden layer output is projected to a lower dimension before calculating dispersion scores, which speeds up the dispersion score calculation with high-dimensional output.

IV. Experimental Results

We evaluate BATER on the following well-known image classification datasets: MNIST [83], CIFAR10 [84] and Imagenet-sub [85]. The training sets provided by the datasets are used to train BNN and DNN. The BNN and DNN architectures are the same, except that the weights of BNN follow Gaussian distributions. We train BNN with Gaussian variational inference because it is straightforward to implement. We have also tried to train BNN with other techniques, such as K-FAC [86], but they all generate similar results.

The test sets are split into 20% in training folds and 80% in test folds. The detection models (binary classifiers) of KD, LID and BATER are trained on the training folds and the test folds are used to evaluate the performance of different detection methods. Foolbox [87] is used to generate adversarial examples with the following attack methods: FGSM [1] with ℓ_∞ norm, PGD [17] with ℓ_∞ norm and C & W [35] with ℓ_2 norm. Since BNN is stochastic, original PGD and C & W attacks without considering randomness are not strong enough against it. For fair comparison, we update PGD and C & W with stochastic optimization methods (multiple forward passes are used to estimate gradient not just one pass).

Experiments in Sections IV-A to IV-E are done in a gray-box setting, in which we assume the adversary has access to the classifier model but does not know the detector. An adaptive attack is proposed in Section IV-F to attack BATER in a white-box setting, in which we assume the adversary has access to both the classifier and the detector. Details of parameter selection, neural network architectures, implementation, code github and examples of detected adversarial examples are provided in the Appendix.

A. Comparison With State-of-the-Art Methods

We compare the performance of BATER with the following state-of-the-art detection methods for adversarial detection: 1) Kernel Density Detection (KD) [19], 2) Local Intrinsic Dimensionality detection (LID) [13], 3) Odds are Odd Detection (ODD) [74], 4) Joint statistical Testing across DNN Layers for Anomalies (JTLA) [75]. In [75], JTLA outperforms deep Mahalanobis detection [23], deep KNN [88], and trust score [89], so we do not include the performance of the three here. Details of implementation and parameters can be found in the Appendix. All the detection methods are tested by the following attacks: 1) FGSM [1] with ℓ_∞ norm bounded by 0.3, 0.03 and 0.01 for MNIST, CIFAR10 and Imagenet-sub respectively; 2) PGD [17] with ℓ_∞ norm bounded by 0.3, 0.03 and 0.01 for MNIST, CIFAR10 and Imagenet-sub respectively; C & W [35] with confidence of 0 for all three datasets.

We report the AUC (Area Under the receiver operating characteristic Curve) score as the performance evaluation criterion as well as the True Positive Rates (TPR) by thresholding False Positive Rates (FPR) at 0.01, 0.05 and 0.1, as it is practical to keep mis-classified natural data at a low proportion. TPR represents the proportion of adversarial examples classified as adversarial, and FPR represents the proportion of natural data mis-classified as adversarial. Before calculating performance metrics, all the adversarial examples that can be classified correctly by the model are removed. The results are reported in Table I and ROC curves are shown in Fig. 4. BATER shows superior or comparable performance over the other four detection methods across three datasets against three attacks.

B. Ablation Study: BNN versus DNN

In this section, we compare the performance of BATER using different structures (BNN versus DNN) against PGD across three datasets. The ℓ_∞ norm is bounded by 0.3, 0.03 and 0.01 for MNIST, CIFAR10 and Imagenet-sub respectively. The detection methods are the same (as described in Algorithm 1) and the differences are: 1) BATER with DNN uses a pre-trained deep neural network of the same structure without random components; 2) The number of passes is one as DNN does not produce different outputs with the same input. We report the class conditional AUC of the two different structures across three datasets.

The comparison results on CIFAR10 and MNIST are shown in Table II and the results on Imagenet-sub are shown in Fig. 5. Since there are 143 classes in Imagenet-sub, it is not reasonable to show the results in a table. Instead, we show the AUC histograms of BATER with different structures in Fig. 5. Comparing the AUCs of applying BATER with BNN and DNN on CIFAR10 and MNIST, it is obvious that the BNN structure demonstrates superior performance all the time. On Imagenet-sub, the AUC histogram of BATER with BNN ranges from 0.90 to 1.00 and is left-tailed, while the AUC histogram of BATER with DNN ranges from 0.10 to 0.85 and centers around 0.40, so the BNN structure clearly outperforms on Imagenet-sub. The experimental results show that random components can help improve detection results.

C. Transfer Attack

In this section, we study the performance of BATER under transfer attack setting. In practice, the defense method does not know what attack methods will be used. Therefore, defense methods trained with adversarial examples generated from one attack method may be attacked by adversarial examples generated by another attack method. When generating adversarial examples, we employ the same attack parameters as outlined in

TABLE I
PERFORMANCE OF DETECTION METHODS AGAINST ADVERSARIAL ATTACKS

Data	Metric	C&W					FGSM					PGD				
		KD	LID	ODD	JTLA	BATER	KD	LID	ODD	JTLA	BATER	KD	LID	ODD	JTLA	BATER
CIFAR10	AUC	0.945	0.947	0.955	0.968	**0.980**	0.873	0.957	0.968	0.990	**0.995**	0.791	0.777	0.963	0.962	**0.971**
	TPR(FPR@0.01)	0.068	0.220	0.591	0.309	**0.606**	0.136	0.385	0.224	0.698	**0.878**	0.018	0.093	0.059	0.191	**0.813**
	TPR(FPR@0.05)	0.464	0.668	0.839	0.726	**0.881**	0.401	0.753	0.709	0.974	**0.991**	0.148	0.317	0.819	0.789	**0.881**
	TPR(FPR@0.10)	0.911	0.856	0.901	0.954	**0.965**	0.572	0.875	**1.000**	**1.000**	0.998	0.285	0.448	**0.999**	**0.999**	0.917
MNIST	AUC	0.932	0.785	0.968	0.980	**0.999**	0.933	0.888	0.952	0.992	**0.999**	0.801	0.861	0.967	0.975	**0.989**
	TPR(FPR@0.01)	0.196	0.079	0.212	0.630	**0.974**	0.421	0.152	0.898	0.885	**0.972**	0.062	0.170	0.607	0.382	**0.733**
	TPR(FPR@0.05)	0.616	0.263	0.911	0.900	**0.997**	0.692	0.503	0.908	0.990	**0.998**	0.275	0.396	0.934	0.851	**0.957**
	TPR(FPR@0.10)	0.818	0.397	**1.000**	0.972	**1.000**	0.796	0.678	0.917	**1.000**	**1.000**	0.429	0.552	0.945	0.956	**0.999**
Imagenet -sub	AUC	0.811	0.905	0.886	0.834	**0.941**	0.914	0.983	0.844	0.842	**0.989**	0.989	**0.991**	0.777	0.824	0.976
	TPR(FPR@0.01)	0.193	**0.401**	0.185	0.035	0.146	0.460	**0.772**	0.042	0.045	0.569	**0.930**	0.829	0.010	0.028	0.729
	TPR(FPR@0.05)	0.452	**0.653**	0.398	0.167	0.538	0.727	0.952	0.188	0.197	**0.989**	**0.966**	0.961	0.054	0.139	0.904
	TPR(FPR@0.10)	0.584	0.754	0.566	0.312	**0.815**	0.822	0.987	0.364	0.358	**1.000**	0.979	**0.984**	0.121	0.280	0.947

The best performance among the five detection methods is marked in bold. In general, BATER performs the best or comparable to the best in most cases.

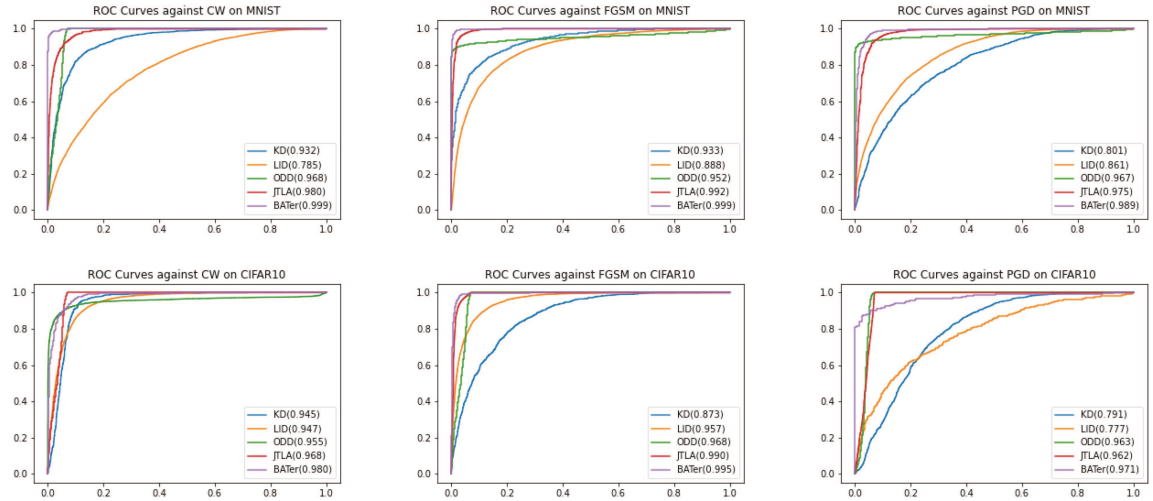
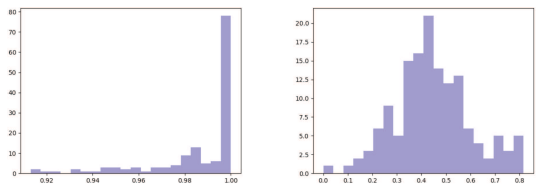


Fig. 4. ROC Curves of experiments in Section IV-A on MNIST and CIFAR10. The curves show that BATER outperforms other detection methods or perform comparably to the best method in all the cases.

TABLE II
AUCS OF BATER WITH DIFFERENT STRUCTURES (BNN VS. DNN) ON CIFAR10 AND MNIST OF DIFFERENT CLASSES. SINCE IN ALL THE CASES, BNN GIVES BETTER RESULTS, IT IS CLEAR THAT BNN IS A BETTER CHOICE THAN DNN, WHICH SHOWS THAT RANDOM COMPONENTS CAN HELP IMPROVE DETECTION PERFORMANCE

Class	CIFAR10		MNIST	
	BNN	DNN	BNN	DNN
class1	0.978	0.489	0.929	0.901
class2	0.972	0.410	1.000	0.967
class3	0.973	0.501	0.993	0.892
class4	0.994	0.594	0.991	0.958
class5	0.955	0.477	1.000	0.883
class6	0.995	0.729	0.999	0.937
class7	0.976	0.584	0.989	0.878
class8	0.973	0.537	1.000	0.941
class9	0.915	0.493	0.959	0.874
lass10	0.949	0.567	0.982	0.917



(a) AUC of BNN (b) AUC of DNN

Fig. 5. AUC Histograms of BATER with different structures (BNN vs. DNN) on Imagenet-sub. It is obvious that BNN results in better AUCs.

TABLE III
PERFORMANCE OF BATER UNDER TRANSFER ATTACK. THE COLUMN NAMES REPRESENT THE ADVERSARIAL EXAMPLES THE DETECTOR TRAINED WITH. THE ROW NAMES REPRESENT THE ADVERSARIAL EXAMPLES THE DETECTOR TESTED AGAINST. AUC SCORES ARE REPORTED

Data	MNIST			CIFAR10			Imagenet-sub		
	C&W	PGD	FGSM	C&W	PGD	FGSM	C&W	PGD	FGSM
C&W	0.999	0.994	0.994	0.980	0.877	0.972	0.941	0.845	0.870
PGD	0.989	0.989	0.989	0.820	0.971	0.824	0.886	0.976	0.975
FGSM	0.998	0.998	0.999	0.868	0.912	0.995	0.914	0.896	0.989

Section IV-A.The performance of BATER in the transfer attack setting are shown in Table III. The results show that BATER trained on one type of adversarial examples can generalize to other types.

D. *Effect of Number of Forward Pass*

The proposed method is based on two blocks: 1) The first part is that the distributional difference between natural/adversarial images of BNN is larger compared to that of DNN. Unfortunately, we cannot prove this part theoretically, but observe the phenomenon empirically (e.g., Fig. 3). 2) Proposition 1 shows that this distributional difference can be enlarged by leveraging the randomness of the BNN model (through multiple passes). Ideally, we need to generate distributions from an infinite number of passes, which is impossible in real practice. Therefore, we

TABLE IV
EFFECT OF THE NUMBER OF FORWARD PASSES ON MNIST AGAINST PGD ATTACK. INCREASING THE NUMBER OF PASSES IS HELPFUL BUT A VERY LARGE NUMBER IS NOT NECESSARY AS 4 PASSES ALREADY SHOWS REASONABLY GOOD RESULTS

num of pass	1	4	6	8	10
AUC	0.9651	0.9892	0.9918	0.9908	0.9897
tpr(fpr@0.01)	0.5471	0.7333	0.7891	0.8051	0.7801
tpr(fpr@0.05)	0.8362	0.9569	0.9709	0.9720	0.9654
tpr(fpr@0.10)	0.9013	0.9990	0.9994	0.9903	0.9865

TABLE V
PERFORMANCE OF DETECTION METHODS AGAINST ADVERSARIAL ATTACKS WITH DIFFERENT PARAMETERS. OUT OF 27 AUC VALUES, 24 OF THEM ARE ABOVE 0.980 AND ALL THE AUCS ARE ABOVE 0.920. BATER PERFORMS WELL AGAINST ATTACKS OF DIFFERENT STRENGTHS

Data	Metric/Parameter	C&W			FGSM			PGD		
CIFAR10	Parameter Value	0	10	20	0.01	0.03	0.05	0.01	0.03	0.05
	AUC	0.980	0.999	0.995	0.982	0.995	0.996	0.965	0.971	0.981
	TPR(FPR@0.01)	0.606	0.998	0.939	0.497	0.878	0.839	0.287	0.813	0.834
	TPR(FPR@0.05)	0.881	1.000	0.995	0.942	0.991	0.996	0.917	0.881	0.928
	TPR(FPR@0.10)	0.965	1.000	0.995	0.978	0.998	0.996	0.960	0.917	0.957
MNIST	Parameter Value	0	10	20	0.1	0.3	0.5	0.1	0.3	0.5
	AUC	0.999	0.995	0.995	0.993	0.999	0.999	0.980	0.989	0.996
	TPR(FPR@0.01)	0.974	0.913	0.919	0.817	0.972	1.000	0.692	0.733	0.920
	TPR(FPR@0.05)	0.997	0.993	0.994	0.994	0.998	1.000	0.973	0.957	0.992
	TPR(FPR@0.10)	1.000	0.998	0.999	0.998	1.000	1.000	0.992	0.999	0.996
Imagenet -sub	Parameter Value	0	10	20	0.01	0.02	0.03	0.01	0.02	0.03
	AUC	0.941	0.991	0.983	0.989	0.992	0.994	0.976	0.982	0.987
	TPR(FPR@0.01)	0.146	0.896	0.642	0.569	0.824	0.841	0.729	0.511	0.708
	TPR(FPR@0.05)	0.538	0.951	0.910	0.989	0.985	0.995	0.904	0.936	0.951
	TPR(FPR@0.10)	0.815	0.977	0.964	1.000	0.997	0.999	0.947	0.980	0.984

conducted experiments to study the effect of the number of forward passes on MNIST against PGD attack. The ℓ_∞ norm of PGD attack is bounded by 0.3 in the experiments.

As shown in Table IV, a few passes can recover this property. Comparing the performance of 4 passes and 1 pass, we see that increasing the number of passes helps improve performance. However, after a certain point, this increase does not improve the performance much. Therefore, we do not need to worry that too many forward passes will be required for the distribution simulation.

E. Defense Against Attack With Different Parameters

Some previous works [11] pointed out that detection methods can fail when the adversarial attacks are strong, such as C & W attack with high confidence. Therefore, we test BATER against adversarial attacks of different strengths across three datasets. For PGD and FGSM attacks, the parameter ϵ captures the strength of the attack with larger ϵ representing a stronger attack. For C & W, we try different confidence levels. The performance of BATER is reported in Table V. Out of 27 AUC values, 24 of them are above 0.980 and all the AUCs are above 0.920. The results show that BATER performs well against various adversarial attacks with different strengths.

F. Adaptive Attack

All the previous experiments are carried out in a gray-box setting, where we assume the adversary has access to the classifier model but does not know the details of the detector. The white-box setting assumes that the adversary has access to both the classifier and the detector. Therefore, an adaptive attack method can be built to attack both the classifier and the detector. This is worth studying as it can reveal possible drawbacks of the method and promote future research direction.

TABLE VI
PEFORMANCE OF BATER AGAINST ADAPTIVE ATTACK. CONSIDERING BOTH ROBUST ACCURACY AND DETECTION AUC, BATER SHOWS ACCEPTABLE PERFORMANCE AGAINST THE ADAPTIVE ATTACK

Metric	MNIST	CIFAR10	Imagenet-sub
Robust.Acc	0.203	0.112	0.215
AUC	0.644	0.801	0.583
TPR(FPR@0.01)	0.325	0.134	0.051
TPR(FPR@0.05)	0.432	0.346	0.171
TPR(FPR@0.10)	0.476	0.450	0.256

To develop an adaptive attack against BATER, we propose the following objective:

$$\underset{\|\boldsymbol{x}-\boldsymbol{x}_0\|_\infty \le \epsilon}{\operatorname{argmin}} \; -L_1(\boldsymbol{x}, y_0) - \lambda L_2(\boldsymbol{x}, z_0), \tag{6}$$

where L_1 and L_2 represent the classification loss and detection loss respectively, λ controls the trade-off between the two, y_0 is the label of original input, z_0 is the detection label, and $\boldsymbol{x}$ and $\boldsymbol{x}_0$ represent adversarial example and original input. The loss function aims to fool the classifier and the detector at the same time. In the experiment, we set $\lambda = 1$. To optimize over the loss function, we build a torch version of the Wasserstein distance function based on the one from the scipy package, making it possible to get the gradient of the second part of the loss function. Due to the sorting operations in the Wasserstein distance calculation, the function is non-differentiable at some points. However, if we are not at those points we can assume the permutation won't change within a small region, so it becomes differentiable using the same permutation forward and backward. So, the gradient is still an approximation but very close.

The performance of BATER against the adaptive attack on 1000 randomly selected images of each dataset is shown in Table VI. We employ the same attack parameters as outlined in Section IV-A. Compared to the gray-box setting, the performance drops, but still reasonable and better than without the detection system. The task of fooling the detection part makes the robust accuracy increase. On MNIST, the robust accuracy increases to 20.3% and the AUC drops to 0.644. Taking both robust accuracy and detection AUC into consideration, the framework can still handle a reasonable portion of adversarial examples correctly. On CIFAR10, though the robust accuracy only increases to 11.2%, the detection AUC is 0.801. On Imagenet-sub, the performance is similar to that on MNIST.

V. CONCLUSION

In this paper, we introduce a new framework to detect adversarial examples with Bayesian Neural Network, by capturing the distributional differences of multiple hidden layer outputs between the natural and adversarial examples. We show that our detection framework outperforms other state-of-the-art methods in detecting adversarial examples generated by various kinds of attacks. It also displays strong performance in detecting adversarial examples generated by various attack methods with different strengths and adversarial examples generated by an adaptive attack method.

REFERENCES

[1] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. Int. Conf. Learn. Representations*, 2015.

[2] C. Szegedy et al., "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*.

[3] V. Kurkova, Y. Manolopoulos, B. Hammer, L. Iliadis, and I. Maglogiannis, *Artif. Neural Netw. Mach. Learn.: 27th Int. Conf. Artif. Neural Netw.*, 2018.

[4] P. Yang, *Towards Adversarial Robustness of Deep Neural Networks*. Davis, CA, USA: Univ. California, 2020.

[5] Y. Li, *On Robustness and Efficiency of Machine Learning Systems*. Davis, CA, USA: Univ. California, 2020.

[6] Y. Li, M. Cheng, C.-J. Hsieh, and T. C. Lee, "A review of adversarial attack and defense for classification methods," *Amer. Statistician*, vol. 76, no. 4, pp. 329–345, 2022.

[7] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2805–2824, Sep. 2019.

[8] J. Zhang and C. Li, "Adversarial examples: Opportunities and challenges," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 7, pp. 2578–2593, Jul. 2020.

[9] A. Chan et al., "Breaking neural reasoning architectures with metamorphic relation-based adversarial examples," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 11, pp. 6976–6982, Nov. 2022.

[10] Y. Li, M. Cheng, C.-J. Hsieh, and T. C. M. Lee, "A review of adversarial attack and defense for classification methods," *Amer. Statistician*, vol. 76, pp. 329–345, 2022, doi: 10.1080/00031305.2021.2006781.

[11] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 274–283.

[12] G. S. Dhillon et al., "Stochastic activation pruning for robust adversarial defense," in *Proc. Int. Conf. Learn. Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=H1uR4GZRZ

[13] X. Ma et al., "Characterizing adversarial subspaces using local intrinsic dimensionality," in *Proc. Int. Conf. Learn. Representations*, 2018.

[14] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-GAN: Protecting classifiers against adversarial attacks using generative models," in *Proc. Int. Conf. Learn. Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=BkJ3ibb0-

[15] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, "PixelDefend: Leveraging generative models to understand and defend against adversarial examples," in *Proc. Int. Conf. Learn. Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rJUYGxbCW

[16] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille, "Mitigating adversarial effects through randomization," in *Proc. Int. Conf. Learn. Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=Sk9yuql0Z

[17] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. Int. Conf. Learn. Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rJzIBfZAb

[18] T. Tanay and L. Griffin, "A boundary tilting persepective on the phenomenon of adversarial examples," 2016, *arXiv:1608.07690*.

[19] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," 2017, *arXiv:1703.00410*.

[20] Z. Zheng and P. Hong, "Robust detection of adversarial attacks by modeling the intrinsic properties of deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 7913–7922.

[21] T. Pang, C. Du, Y. Dong, and J. Zhu, "Towards robust detection of adversarial examples," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 4584–4594.

[22] G. Tao, S. Ma, Y. Liu, and X. Zhang, "Attacks meet interpretability: Attribute-steered detection of adversarial samples," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 7717–7728.

[23] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 7167–7177.

[24] P. Yang, J. Chen, C.-J. Hsieh, J.-L. Wang, and M. Jordan, "ML-LOO: Detecting adversarial examples with feature attribution," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 6639–6647.

[25] X. Ding, Y. Cheng, Y. Luo, Q. Li, and P. Gope, "Consensus adversarial defense method based on augmented examples," *IEEE Trans. Ind. Inform.*, vol. 19, no. 1, pp. 984–994, Jan. 2023.

[26] X. Liu, M. Cheng, H. Zhang, and C.-J. Hsieh, "Towards robust neural networks via random self-ensemble," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 369–385.

[27] J. M. Cohen, E. Rosenfeld, and J. Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *Proc. Int. Conf. Mach. Learn.*,2019, pp. 1310–1320.

[28] X. Liu, Y. Li, C. Wu, and C.-J. Hsieh, "Adv-BNN: Improved adversarial defense through robust Bayesian neural network," in *Proc. Int. Conf. Learn. Representations*, 2019.

[29] N. Ye and Z. Zhu, "Bayesian adversarial learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 6892–6901. [Online]. Available: http://papers.nips.cc/paper/7921-bayesian-adversarial-learning.pdf

[30] G. Carbone, M. Wicker, L. Laurenti, A. Patane, L. Bortolussi, and G. Sanguinetti, "Robustness of Bayesian neural networks to gradient-based attacks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 15602–15613.

[31] C.-H. H. Yang et al., "Mitigating closed-model adversarial examples with Bayesian neural modeling for enhanced end-to-end speech recognition," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2022, pp. 6302–6306.

[32] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1613–1622.

[33] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9 , pp. 2805–2824, Sep. 2019.

[34] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, 2017, pp. 3–14.

[35] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 39–57.

[36] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1765–1773.

[37] Z. Che et al., "SMGEA: A new ensemble adversarial attack powered by long-term gradient memories," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 3, pp. 1051–1065, Mar. 2022.

[38] L. Liang et al., "Exploring adversarial attack in spiking neural networks with spike-compatible gradient," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 5, pp. 2569–2583, May 2023.

[39] P. Zhao, K. Xu, S. Liu, Y. Wang, and X. Lin, "ADMM attack: An enhanced adversarial attack for deep neural networks with undetectable distortions," in *Proc. 24th Asia South Pacific Des. Automat. Conf.*, 2019, pp. 499–505.

[40] T. Chen, J. Liu, Y. Xiang, W. Niu, E. Tong, and Z. Han, "Adversarial attack and defense in reinforcement learning-from AI security view," *Cybersecurity*, vol. 2, pp. 1–22, 2019.

[41] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2574–2582.

[42] P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh, "EAD: Elastic-net attacks to deep neural networks via adversarial examples," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, Art. no. 2.

[43] N. Carlini, *Evaluation and Design of Robust Neural Network Defenses*. Berkeley, CA, USA: Univ. California, 2018.

[44] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, 2017, pp. 15–26.

[45] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," in *Proc. Int. Conf. Learn. Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=SyZI0GWCZ

[46] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2137–2146.

[47] M. Cheng, S. Singh, P.-Y. Chen, S. Liu, and C.-J. Hsieh, "Sign-OPT: A query-efficient hard-label adversarial attack," in *Proc. Int. Conf. Learn. Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=SklTQCNtvS

[48] Z. Yan, Y. Guo, and C. Zhang, "Subspace attack: Exploiting promising subspaces for query-efficient black-box attacks," *Adv. Neural Inf. Process. Syst.*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. New York, NY, USA: Curran Associates, Inc., vol. 32, 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/2cad8fa47bbef282badbb8de5374b894-Paper.pdf

[49] J. Chen, M. I. Jordan, and M. J. Wainwright, "HopSkipJumpAttack: A query-efficient decision-based attack," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 1277–1294.

[50] J. Chen, *Towards Interpretability and Robustness of Machine Learning Models*. Berkeley, CA, USA: Univ. California, 2019.

[51] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *Artif. Intell. Saf. Secur.*, pp. 99–112, 2018.

[52] F. Tramàr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," in *Proc. Int. Conf. Learn. Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rkZvSe-RZ

[53] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7472–7482.

[54] N. Ye, Q. Li, X.-Y. Zhou, and Z. Zhu, "An annealing mechanism for adversarial training acceleration," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 2 , pp. 882–893, Feb. 2023.

[55] D. Meng and H. Chen, "MagNet: A two-pronged defense against adversarial examples," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 135–147.

[56] Y. Li et al., "Towards robustness of deep neural networks via regularization," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2021, pp. 7496–7505.

[57] A. Jalal, A. Ilyas, C. Daskalakis, and A. G. Dimakis, "The robust manifold defense: Adversarial training using generative models," 2017, *arXiv:1712.09196*.

[58] Y. Li et al., "Towards robustness of deep neural networks via regularization," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 7496–7505.

[59] E. Wong and Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5286–5295.

[60] M. Everett, B. Lütjens, and J. P. How, "Certifiable robustness to adversarial state uncertainty in deep reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 9, pp. 4184–4198, Sep. 2022.

[61] X. Chen, J. Weng, X. Deng, W. Luo, Y. Lan, and Q. Tian, "Feature distillation in deep attention network against adversarial examples," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 7, pp. 3691–3705, Jul. 2023.

[62] Q. Liu and W. Wen, "Model compression hardens deep neural networks: A new perspective to prevent adversarial attacks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 1, pp. 3–14, Jan. 2023.

[63] B. Zhang, B. Tondi, X. Lv, and M. Barni, "Challenging the adversarial robustness of DNNs based on error-correcting output codes," *Secur. Commun. Netw.*, vol. 2020, pp. 1–11, 2020.

[64] A. Mustafa, S. H. Khan, M. Hayat, R. Goecke, J. Shen, and L. Shao, "Deeply supervised discriminative learning for adversarial defense," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 9, pp. 3154–3166, Sep. 2021.

[65] S. Zhao, J. Yu, Z. Sun, B. Zhang, and X. Wei, "Enhanced accuracy and robustness via multi-teacher adversarial distillation," in *Proc. 17th Eur. Conf. Comput. Vis.*, 2022, pp. 585–602.

[66] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," in *Proc. Int. Conf. Learn. Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=SJzCSf9xg

[67] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," in *Proc. Int. Conf. Learn. Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=SJzCSf9xg

[68] A. Agarwal, G. Goswami, M. Vatsa, R. Singh, and N. K. Ratha, "DAMAD: Database, attack, and model agnostic adversarial perturbation detector," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 8 , pp. 3277–3289, Aug. 2022.

[69] F. Nesti, A. Biondi, and G. Buttazzo, "Detecting adversarial examples by input transformations, defense perturbations, and voting," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 3, pp. 1329–1341, Mar. 2023.

[70] Z. Gong and W. Wang, "Adversarial and clean data are not twins," in *Proc. 6th Int. Workshop Exploiting Artif. Intell. Techn. Data Manage.*, 2023, pp. 1–5.

[71] P. Sperl, C.-Y. Kao, P. Chen, X. Lei, and K. Böttinger, "DLA: Dense-layer-analysis for adversarial example detection," in *Proc. IEEE Eur. Symp. Secur. Privacy*, 2020, pp. 198–215.

[72] S. Gao et al., "Detecting adversarial examples on deep neural networks with mutual information neural estimation," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 6 , pp. 5168–5181, Nov./Dec. 2023.

[73] Y. Chen, M. Zhang, J. Li, and X. Kuang, "Adversarial attacks and defenses in image classification: A practical perspective," in *Proc. IEEE 7th Int. Conf. Image Vis. Comput.*, 2022, pp. 424–430.

[74] K. Roth, Y. Kilcher, and T. Hofmann, "The odds are odd: A statistical test for detecting adversarial examples," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 5498–5507.

[75] J. Raghuram, V. Chandrasekaran, S. Jha, and S. Banerjee, "A general framework for detecting anomalous inputs to DNN classifiers," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 8764–8775.

[76] Y. Gao, Z. Lin, Y. Yang, and J. Sang, "Towards black-box adversarial example detection: A data reconstruction-based method," 2023, *arXiv:2306.02021*.

[77] Y. Zhou, J.-Y. Jiang, K.-W. Chang, and W. Wang, "Learning to discriminate perturbations for blocking adversarial attacks in text classification," in *Proc. Conf. Empir. Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process.*, 2019, pp. 4903–4912.

[78] M. Mozes, P. Stenetorp, B. Kleinberg, and L. Griffin, "Frequency-guided word substitutions for detecting textual adversarial examples," in *Proc. 16th Conf. Eur. Chapter Assoc. Comput. Linguistics*, 2021, pp. 171–186. [Online]. Available: https://aclanthology.org/2021.eacl-main.13

[79] K. Yoo, J. Kim, J. Jang, and N. Kwak, "Detection of adversarial examples in text classification: Benchmark and baseline via robust density estimation," in *Proc. Findings Assoc. Comput. Linguistics*, 2022, pp. 3656–3672. [Online]. Available: https://aclanthology.org/2022.findings-acl.289

[80] F. Yin, Y. Li, C.-J. Hsieh, and K.-W. Chang, "ADDMU: Detection of far-boundary adversarial examples with data and model uncertainty estimation," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2022, pp. 6567–6584.

[81] H. Ren, T. Huang, and H. Yan, "Adversarial examples: Attacks and defenses in the physical world," *Int. J. Mach. Learn. Cybern.*, vol. 12, pp. 3325–3336, 2021.

[82] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[83] Y. LeCun, "The MNIST database of handwritten digits," 1998. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[84] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Toronto, ON, Canada, 2009.

[85] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *Proc. Int. Conf. Learn. Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=B1QRgziT-

[86] G. Zhang, S. Sun, D. Duvenaud, and R. Grosse, "Noisy natural gradient as variational inference," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5852–5861.

[87] J. Rauber, R. Zimmermann, M. Bethge, and W. Brendel, "Foolbox native: Fast adversarial attacks to benchmark the robustness of machine learning models in PyTorch, TensorFlow, and JAX," *J. Open Source Softw.*, vol. 5, no. 53, p. 2607, 2020, doi: 10.21105/joss.02607.

[88] N. Papernot and P. McDaniel, "Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning," 2018, *arXiv:1803.04765*.

[89] H. Jiang, B. Kim, M. Y. Guan, and M. R. Gupta, "To trust or not to trust a classifier," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 5546–5557.

Yao Li received the bachelor's degree in statistics from Fudan University, Shanghai, China, in 2014, and the Ph.D. degree from the University of California, Davis, Davis, CA, USA, in 2020. She is currently an Assistant Professor of statistics and operations research with the University of North Carolina at Chapel Hill, Chapel Hill, NC, USA. Her research interests include trustworthy machine learning, computational pathology, and machine learning applications in other scientific disciplines.

Tongyi Tang received the bachelor's degree in mathematics from Fudan University, Shanghai, China, in 2016, and the Ph.D. degree from the University of California, Davis, Davis, CA, USA, in 2021. She is currently a Research Scientist with Meta Platforms, Inc., Menlo Park, CA. Her research interests include optimization, random vector field modeling, and security of deep learning models.

Cho-Jui Hsieh is currently an Associate Professor with the Computer Science Department, University of California, Los Angeles, Los Angeles, CA, USA. His work primarily focuses on enhancing the efficiency and robustness of machine learning systems. He has made significant contributions to multiple widely-used machine learning packages. He was the recipient of the NSF Career Award, Samsung AI Researcher of the Year, and Google Research Scholar Award, and his work has been acknowledged with several paper awards in ICLR, KDD, ICDM, ICPP, and SC.

Thomas C. M. Lee (Senior Member, IEEE) received the B.App.Sc. and B.Sc. (Hons) (with University Medal) degrees in mathematics from the University of Technology, Sydney, NSW, Australia, in 1992 and 1993, respectively, and the Ph.D. degree jointly from Macquarie University, Macquarie Park, NSW, and CSIRO Mathematical and Information Sciences, Sydney, in 1997. He is currently a Professor of statistics and an Associate Dean of the Faculty of Mathematical and Physical Sciences, University of California, Davis (UC Davis), Davis, CA, USA. His research interests include inference methods, machine learning, and statistical applications in other scientific disciplines. He is an elected Fellow of the American Association for the Advancement of Science, American Statistical Association, and Institute of Mathematical Statistics. From 2013 to 2015, he was the Editor-in-Chief of the *Journal of Computational and Graphical Statistics*, from 2015 to 2018, and the Chair of the Department of Statistics, UC Davis. He is also the Review Editor of the *Journal of the American Statistical Association*.