Achieving Both Model Accuracy and Robustness by Adversarial Training with Batch Norm Shaping

Brian Zhang
Harrison High School (Tippecanoe)
bzhangprogramming@gmail.com

Shiqing Ma
Rutgers University
sm2283@cs.rutgers.edu

***Abstract*—Adversarial training is an important approach to improving Deep Learning model robustness. It uses attack methods to generate adversarial samples that can maximize the chance of misclassification and updates model weight values accordingly to ensure these samples are not misclassified. It is difficult to retain model accuracy while improving robustness using adversarial training. In this paper, we study one of the important factors causing this undesirable effect - batch normalization. We find that batch normalization has three confoundings in adversarial training, which may cause model accuracy degradation and/or sub-optimal robustness improvement. We propose a novel adversarial training method called norm shaping, in which a model always uses batch norms, in both adversarial training and inference. It enforces that a batch (in both training and inference) should always have at least a dominating portion of clean samples such that the batch norms follow a distribution similar to that of clean sample batches. Our results show that it can substantially improve existing adversarial training methods (for models with batch normalization layers), such as PGD and TRADES. On CIFAR-10, it can achieve much better model accuracy and robustness on a list of existing attacks. For example, it can achieve 0.94 model accuracy and 0.81 robustness against PGD attack while TRADES and PGD adversarial trainings can achieve around 0.88 accuracy and 0.47 robustness. Our method also has 0.51 robustness against the strongest adaptive attack.**

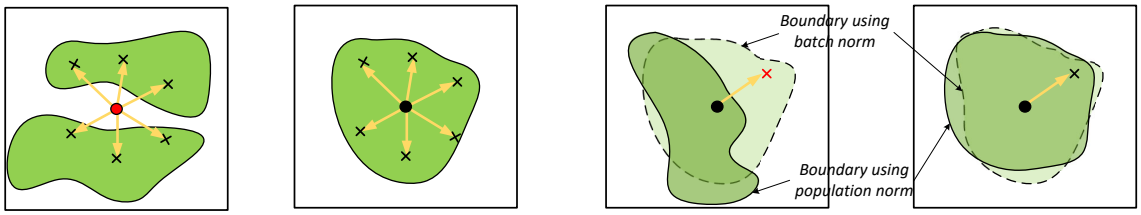
***Index Terms*—robustness, adversarial training, batch normalization, PGD**

I. Introduction

Robustness in Deep Learning dictates that model classification results should be robust in the presence of bounded input perturbation. It is an important property because in real world applications, model inputs have all kinds of noise due to environmental condition variations. Model misbehaviors such as misclassification may be induced if an unrobust model is used. Many may have catastrophic consequences. For example, perception model misclassification (e.g., object detection model or depth estimation model) in autonomous driving vehicles may endanger human lives. There are many methods to improve model robustness and trustability of classification results even when the model is unrobust, such as adversarial input detection [6], [10], [21], [24], [36], model certification [16], model symbolic analysis [3], [9], [12], [19], [33], [37], and adversarial training [13], [18], [23], [26], [29], [30], [32], [38]. T Among them, adversarial training is one of the most popular methods. It leverages *adversarial attack* to generate input perturbations for given clean inputs. The perturbed inputs are called *adversarial samples*, which are used to train the model such that misclassifications can be prevented.

A prominent challenge in adversarial training is that model accuracy (on clean samples) degrades when model robustness is improved. For example, with the standard PGD adversarial training [23], which generates adversarial samples by iteratively perturbing inputs following the opposite gradient directions (in order to maximize the chance of inducing misclassification), a ResNet w32-10 model on CIFAR-10 can achieve close to 0.50 robust accuracy (i.e., classification accuracy on adversarial samples). However, its model accuracy degrades from 0.93 to lower than 0.88. Researchers believe that because in many adversarial training methods like PGD, only adversarial samples are used in training and clean samples are not used, clean samples become out-of-distribution, causing model accuracy degradation [4], [35]. Figure 1 (a) provides a conceptual illustration. The small circle in the middle denotes a benign sample and the box denotes its perturbation bound (e.g., pixel value of 8 in PGD). The crosses are the adversarial samples and the yellow arrows denote the perturbations. The green shapes denote *decision boundaries*, including all the data points that are correctly classified. Ideally, the whole box should be green. Observe in figure (a) that since only adversarial samples are used in training, the decision boundary includes all of them. However, the clean sample (in the center) is not included and hence misclassified (denoted by its red color).

There are a number of proposals to train benign and adversarial samples together, such as TRADES [38]. As a result, both kinds of samples are in distribution and a decision boundary like Figure 1 (b) can be achieved, covering both kinds of samples. Although they have achieved very promising results, they still suffer from performance degradation. We observe that the suboptimal results are likely due to the confoundings by normalization. Specifically, during training, statistics of a training batch, called *batch norms* (BN) [15], are used to normalize activation values such that it becomes easier for the training to converge. As a result, the decision boundary is parameterized by the batch norms (and essentially the batch). During inference, the population statistics, called *population norms* (PN), are used in adversarial sample generation and classification. The two norms are different, leading to different decision boundaries [28]. As illustrated by Figure 1 (c), the two shapes denote the two different boundaries with



(a) Training with only adv. samples (b) Training with both clean and adv. samples (c) Decision boundary changes during inference due to norm changes (d) Norm shaping

Fig. 1: Efforts in achieving good tradeoffs between model accuracy and robust accuracy. Boxes denote the perturbation bounds of samples, which are denoted by the circles in the center; crosses denote adversarial samples; green shapes denote decision boundaries (all data points within the boundaries are classified to the ground truth label of samples); and the samples in red are misclassified, that is, they are beyond the boundaries.

the one in dark green by PN. The red cross denotes an adversarial sample generated using PN (at the inference time). Observe that due to the boundary differences, although the sample can be correctly classified by the BN boundary, it is misclassified following the PN boundary (during inference). In some cases, the boundary is changed so substantially that even the clean sample (e.g., the circle in the center of figure (c)) is misclassified. In Section IV, we will discuss three confoundings caused by batch normalization in adversarial training. They can cause model accuracy degradation and/or suboptimal robustness improvement.

In this paper, we propose a technique called *norm shaping* to minimize the norm differences between attack (i.e., adversarial sample generation), training, and inference. As such, the confoundings can be substantially suppressed. The idea is to always use batch norms and in the mean time ensure the stability of norms. Specifically, during attack and training, we enforce a specific ratio between the adversarial and clean samples within a batch, with the latter dominating. That is, during attack (of a batch), only the adversarial samples are perturbed. The dominating number of (unchanged) clean samples ensure that the batch norm and hence the decision boundary are hardly changed across the multiple steps when attacking the batch. This implies that we are attacking the weakest points of a stable decision boundary and then patching the same boundary when updating weights using the (attacked) batch in training. During inference, for each test sample, we group it with a number of (random) clean samples to form a batch with the same ratio, and use batch norms in classification. This ensures that the inference is based on a boundary that the model was trained on. This is illustrated by Figure 1 (d). Observe that since the boundaries are similar, the degradation due to boundary variation is minimized. Here, we assume that in real-world applications, the adversary can access or approximate gradients (for adversarial sample generation). However, he cannot disable norm shaping or determine the set of clean samples used in shaping (as they are randomly drawn for each inference query). This is reasonable as we can trust that the model's execution on the user side is not tampered with. We will have a thorough discussion of our threat model in Section V and study a more aggressive threat model in Section VI. *Our method is complementary to existing adversarial training methods as most of them have such confoundings*. Our contributions are summarized as follows.

- We conduct an in-depth study of the confoundings by batch normalization in adversarial training.
- We propose a novel norm shaping method to boost performance of adversarial training.
- We have evaluated the method on two popular adversarial training methods: PGD [23] and TRADES [38]. It substantially improves their performance, allowing the trained models to achieve model accuracy close to normally trained models, namely, over 0.94 for CIFAR10, and the state-of-the-art robustness against a list of attacks, including PGD, C&W [7], Deepfool [22], and FGSM [13]. For example, it improves the robustness of models trained by PGD from 0.47 to 0.816, and models trained by TRADES from 0.46 to 0.817 (under the PGD attack). It also has around 0.51 robustness against the strongest adaptive attack in which the attacker knows the set of samples used in norm shaping.

Limitations. Our method requires each test sample is packed with a set of clean samples during inference. This enlarges the resource consumption, especially memory footprint. This may limit its applications in scenarios where there are resource constraints. Some may be worried that norm shaping makes inference not independent. We argue that norm shaping samples are randomly drawn from the distribution for each inference request, ensuring independency. In addition, the legitimacy of using batches in inference has been shown in [28], [34]. There are tasks in which batch normalization is not used or adversarial training may not cause performance degradation (e.g., in NLP), our technique is not suitable for them.

II. Background

Adversarial Training. Adversarial training is a widely used method to improve model robustness, which states that a model's prediction be stable in the presence of bounded perturbation. It is often formulated as a minmax optimization problem as follows.

$$\min_{\theta} \mathbb{E}_{(x,y)\sim(\mathcal{X},\mathcal{Y})} \left[\max_{x' \in (x-\epsilon, x+\epsilon)} \mathcal{L}(M(x';\theta), y) \right] \quad (1)$$

Here, a data sample $x \in \mathbb{R}^d$ and its label $y \in \mathbb{N}$ jointly follow distribution $(\mathcal{X}, \mathcal{Y})$. Symbol x' denotes an adversarial sample, M the model with parameter θ, $\mathcal{L}$ the cross-entropy loss function and ϵ a perturbation bound (e.g., in l_∞). Eq. 1 says that the model weights are to minimize the expected maximum cross-entropy loss that bounded adversarial samples can induce. While the formula is declarative, our discussion in the rest of the paper involves its imperative implementation.

Batch Normalization. Batch normalization [15] is a standard technique to make model training more stable and converging faster. It is realized by adding normalization layers to the model such that internal activation values are normalized to mean 0 and standard deviation 1 using the mean and standard deviation of an input batch, called *batch norm* (BN) in this paper. During training, the population mean and standard deviation (over all training samples), called *population norm* (PN) in this paper, are also computed by updating moving mean and standard deviation. During inference, PNs are used to normalize activation values instead of BNs.

III. Related Work

Batch Normalization in Adversarial Training. Researchers in [5], [11] show that BN in natural training may decrease model robustness as it shifts the training to leverage non-robust features (NRF). In [35], Xie and Yuille proposed to improve robustness using alternative batches of clean and adversarial samples in training to mitigate the undesirable effects of batch normalization. The method does not use batch norms during inference. In contrast, we use batch norms all the time and enforce dominating clean samples in a batch. In [4], Benz et al. proposed to rectify BNs using a few attack samples to improve robustness. It is an inference time technique. In [14], [27], researchers show that maintaining two separate norms for clean and adversarial samples can achieve both accuracy and robustness by using the appropriate norms. In contrast, our design has only one norm (achieved by a fixed ratio of adversarial and clean samples), avoiding the need of selecting the right norm during inference. [20] proposed to use a specially designed batch normalization layer to deal with the different norms. Researchers in [28] found that using batch norms can improve both accuracy and robustness and simply co-training clean and adversarial samples together like in TRADES may not work well due to the heterogeneous distributions. Our study supports both observations. [31] proposed to remove batch normalization in adversarial training.

Adversarial Training. There are a large body of existing adversarial training methods based on different attacks, e.g., L-BFGS [29], FGSM [13], one-step [18], PGD [23], TRADES [38], and more [26], [30], [32]. Rice et. al [26] proposed to use early-stop in adversarial training to address overfitting. Tramèr et. al [30] introduced ensemble adversarial training to improve robustness against black-box attacks. Unsupervised adversarial training (UAT) [2] and robust self-training (RST) [8] use additional unlabeled samples to improve robustness. Our method is orthogonal to most these techniques which use batch normalization.

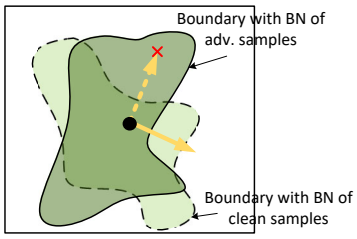


Fig. 2: Norm differences hurt robustness

IV. Confoundings of Batch Normalization in Adversarial Training

Batch normalization causes a number of confoundings in adversarial training. We discuss these confoundings here.

Confounding I: Norm Differences Degrade Model Accuracy. In normal training, model weights are updated based on BNs while PNs are computed gradually by updating moving means and variances [15]. During inference, the trained PNs are used. For naturally trained models, clean test samples are in the same distribution as the training samples, PNs align well with the BNs of test batches, yielding a good decision boundary and hence high accuracy. However, PNs computed in adversarial training may be quite different from the BNs of clean test batches, causing model accuracy degradation. For example, in PGD adversarial training [23], only adversarial samples are used in training, which are generated to cause model misclassification. That is, training batches contain only adversarial samples without any clean samples. The PNs computed hence denote the distribution of adverarial samples instead of that of clean samples. During testing, the decision boundary denoted by the PNs is hence not optimal for (clean) test samples, causing model accuracy degradtion. A well trained CIFAR-10 model usually has over 0.93 accuracy, whereas a model adversarially trained by PGD usually has lower than 0.88 model accuracy. While there may be other reasons behind the degradation, the norm differences is an important one.

Confounding II: Norm Differences Hinder Model Robustness Improvement. Adversarial training is essentially a minmax problem. The inner maximization is to find adversarial samples that expose the *weakest points* of the model and the outer minimization is to patch these points by updating model weights. Eq. 1 is declarative, only describing the intended constraints. The implementation, however, has to be imperative, consisting of a large number of training steps, each processing a batch of samples through the separated min and max operations.

$$\max_{x' \in (x-\epsilon,\ x+\epsilon)} \mathbb{E}_{(x,y)\in B\sim(\mathcal{X},\mathcal{Y})} \left[\mathcal{L}((M_{BN(B)}(x';\theta),\ y)\right] \quad (2)$$

Conceptually, Eq. 2 describes the max step for a batch of samples denoted by B. We use $M_{BN(B)}$ to denote the forward computation of the batch using its batch norm. Essentially, the max step denotes the adversarial sample generation, in which the BNs of the clean samples in B are used to derive the classification results and then the samples are perturbed to

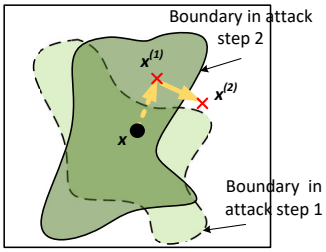


Fig. 3: Batch norms lead to weak attack

maximize the loss. Let x' denote the adversarial sample from x and B' the perturbed batch from B. The minimization step is hence the following.

$$\min_{\theta} \mathbb{E}_{(x',y)\in B'} \left[\mathcal{L}(M_{BN(B')}(x';\theta), y)\right] \tag{3}$$

Observe that in this step, the model uses the BNs of B'. This induces what we call *the moving target effect*, as denoted in Fig. 2. In particular, the two batch norms denote two decision boundaries. In the adversarial sample generation, the dotted yellow arrow denotes the perturbation, leading to the adversarial sample denoted by the red cross. Observe that the perturbation is along the direction closest to the dotted boundary (i.e., easiest to flip the classification result of clean samples). However, in the second step, i.e., the min operation, the BN of the adversarial samples induces a different decision boundary (in dark green), in which the red cross and the dotted arrow no longer disclose the weakness of the boundary. In contrast, the solid yellow arrow does. *The net effect is that the model is not hardened along its weakest directions (when using adversarial samples in training, like the red cross).* Unfortunately, this is not easy to fix in practice as the new decision boundary is unknown until the adversarial samples are generated.

Experiment. We use FGSM [13], which is equivalent to one attack step in PGD, to generate adversarial samples on training inputs for an adversarially trained model with 0.87 accuracy and 0.47 robustness (using the attack step size of 2). The logits errors between the adversarial samples and their clean versions when the BNs of clean samples are used are on average 63% larger than those when the BNs of adversarial samples are used, indicating that although the adversarial samples reflect the weakness of decision boundary by the BNs of clean samples, they do not reflect the weakness of boundary by the BNs of themselves. □

Confounding III: Attack Using BN Is Weak Due to the Moving Target Effect. In practice, adversarial attack using the max operation defined in Eq. 2 is still too expensive due to its declarative nature. It is hence approximated by an iterative process with a bounded number of attack steps defined in the following.

$$\begin{aligned} x^{(1)} &= (x - \gamma \cdot sign(\Delta_x \mathcal{L}(M_{BN\,(B=\{x\})}(x;\theta), y))) \mid_{(x-\epsilon,\ x+\epsilon)} \\ x^{(2)} &= (x^{(1)} - \gamma \cdot sign(\Delta_{x^{(1)}} \mathcal{L}(M_{BN\,(B^{(1)}=\{x^{(1)}\})}(x^{(1)};\theta), y))) \mid_{(x-\epsilon,\ x+\epsilon)} \\ &\cdots \end{aligned} \tag{4}$$

Specifically, the adversarial samples after step one, denoted by $x^{(1)}$, are perturbed along the opposite direction of gradients using the BNs of clean samples (e.g., x), controlled by the step size γ and delimited by the bound ϵ. The samples after step two, denoted by $x^{(2)}$, are derived from $x^{(1)}$ using the BNs of $x^{(1)}$'s, and so on.

It is widely believed that the strength of attack is denoted by the bound ϵ, the step size γ, and the number of attack steps. However, norm differences can also substantially affect the attack strength. Fig. 3 provides a conceptual illustration. The two green shapes denote the decision boundaries in attack steps 1 and 2, the arrows denote the perturbations and $x^{(1)}$ and $x^{(2)}$ the two adversarial samples after the respective steps. Observe that although the arrows are both along the weakest directions of the decision boundaries (pointing to the closest points on the boundary lines), the directions change from step to step, forming a zig-zagging line. Such an attack is weaker than one that follows a straight line. That is, the distance between x and $x^{(2)}$ is smaller than the sum of the distances from x to $x^{(1)}$ and from $x^{(1)}$ to $x^{(2)}$. Note that the sum essentially denotes the perturbation in two steps when a straight line is followed. In fact during inference, since PNs (constants) are used across the attack steps, the decision boundary is fixed and the attack is along a straight line (through the steps) and hence much stronger. In PGD, BN based attack (weaker) is used in training and PN based attack (stronger) is used in inference. The hardening effect is hence not maximized during training.

Experiment. We use the default PGD attack settings (i.e., ϵ=8, γ=2, and number of steps is 10) to generate adversarial samples for training inputs using the same adversarially trained model as in the previous experiment. We find that when the PNs are used in generation (like in inference), the input perturbations are on average 17% more substantial than those when BNs are used (like in training). □

V. Norm Shaping in Adversary Training

We propose a norm shaping technique to suppress the aforementioned confoundings and improve adversarial training effectiveness. Figure 4 presents an overview of the technique. Figure (a) presents a training step during the adversarial training and (b) the inference process. The former consists of two phases separated by the vertical dashed line, with phase ① (to the left of the dashed line) denoting t steps of attack to generate adversarial samples ($t = 10$ following the standard PGD attack setting) and phase ② (to the right of the dashed line) denoting model weight updates based on the samples. During attack, from left to right a batch B (of clean samples) is first fed to the model. Since batch norms are used during attack, the decision boundary is hence parameterized by $BN(B)$. The batch B is partitioned to $n+1$ equal parts. For discussion simplicity, we assume it has $n+1$ samples. Samples x_1, ..., x_n, called *norm shaping samples*, are not perturbed during the attack steps and directly copied over from step to step. In contrast, sample x is updated based on the gradients to maximize the cross-entropy loss, that is, to

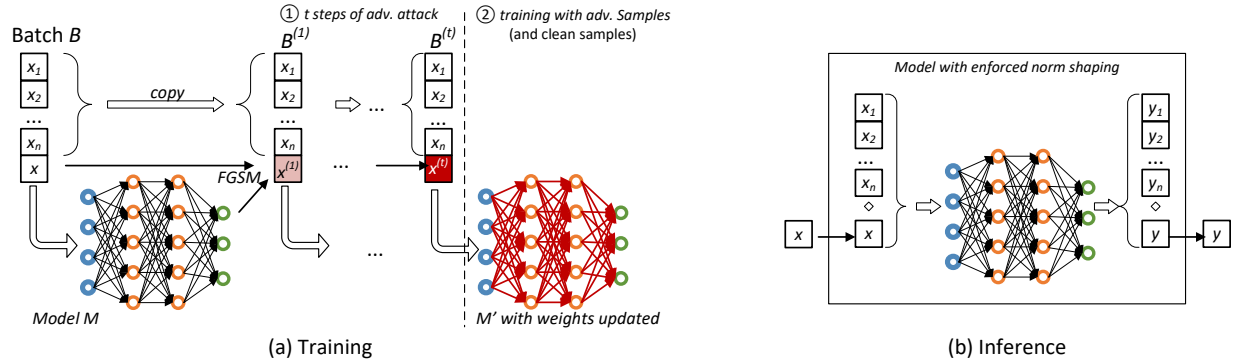


Fig. 4: Norm shaping during adversarial training

induce misclassification just like in the original PGD attack. The new batch $B^{(1)}$ consisting of x_1, ..., x_2, and the updated $x^{(1)}$, is further passed to the model for the next step of attack. After t steps, the final batch $B^{(t)}$ consists of n clean samples and an adversarial sample $x^{(t)}$ approximating the maximum perturbation. The batch is then used in updating the model weights (just like in normal training).

During inference (i.e., Figure 4 (b)), given a sample x, our method forcefully combines it with n clean samples drawn from the training set to form a batch with n:1 ratio between the norm shaping samples and the sample to test. The batch is fed to the model, which operates with the batch norms (instead of the population norms like in typical inference). Only the classification result y of the-sample-to-test is returned.

Design Justification. First, *the model only uses batch norms, in attack, training, and inference, and never uses population norms*. The decision boundary is hence controlled by the norm of given batch. By manipulating the batch, we shape the norms to achieve optimal performance.

Second, *the batches always have the same ratio between clean and adversarial samples, with the former dominating, leading to batch norms closely resembling those of clean samples*. This has multiple ramifications, addressing the aforementioned confoundings.

- With multiple training epochs, the model learns to operate in the batch norms of clean samples as the training batches contain mostly clean samples. During inference, norm shaping enforces similar batch norms, ensuring good accuracy, addressing Confounding I.
- Across the multiple steps of attack on a batch (as part of a training step), the BN is stable due to the stable dominating set of clean samples. As such, the zig-zagging effect is mitigated and the attack becomes stronger, addressing Counfounding III. During training, our attack with norm shaping can induce on average 41% more substantial input perturbations compared to the default PGD attack in the same setting. Recall that PGD uses BNs of adversarial samples.
- The generated adversarial samples provide good approximation of the weakest points of a stable decision boundary (of the current batch with mixed clean and adversarial samples). The weight updates in adversarial training are based on the same boundary and hence enable patching these weakest points, addressing Confounding II.

Threat Model. During inference, we assume the adversary can perturb input x and have read access to the internals of model. This is consistent with the literature. He cannot disable norm shaping or know the norm shaping samples, that is, x_1, ... x_n in Figure 4. This is reasonable as model inference is performed on the user side to which the adversary does not have the write access, and the norm shaping samples are randomly drawn from the input distribution for each inference query. In Section VI, we study a more aggressive threat model in which the adversary knows the set of norm shaping samples.

Formal Definition. Eq. 5 defines an adversarial training step. A training batch B is a pair of X (vector of m samples) and Y (vector of the corresponding labels). Variable k is computed such that x_1, ..., x_k are the set of clean samples, x_{k+1}, ... x_m denote the adversarial samples, and the two are in the ratio of n:1. The $\diamond$ operator concatenates two vectors. Vector $X^{(i)}$ denotes the batch after step i. Observe that the first k samples are unchanged, making the batch norm stable. The remaining samples are updated based on the gradient sign and the step size γ. We want to point out that the method does not completely prevent the moving target effect as the batch norm of $X^{(i)}$ differs from that of $X^{(i+1)}$. However, a large n ensures that the BN differences are small.

$$
\begin{aligned}
&B = (X, Y) = (\{x_1, ...x_m\},\ \{y_1, ...y_m\}),\ (x_i, y_i) \sim (\mathcal{X}, \mathcal{Y}) \\
&\textbf{let } k = \frac{n \cdot m}{n+1} \textbf{ in} \\
&X^{(1)} = X[0:k] \diamond \\
&\quad (X - \gamma \cdot sign(\Delta_X \mathcal{L}(M_{BN(X)}(X;\theta), Y)))[k:m] \mid_{(X-\epsilon,\ X+\epsilon)} \\
&X^{(2)} = X[0:k] \diamond \\
&\quad (X^{(1)} - \gamma \cdot sign(\Delta_{X^{(1)}} \mathcal{L}(M_{BN(X^{(1)})}(X^{(1)};\theta), Y)))[k:m] \mid_{(X-\epsilon,\ X+\epsilon)} \\
&\ldots
\end{aligned}
\tag{5}
$$

Eq. 6 defines the inference. Given an example x, n random samples are drawn from the input distribution (i.e., training samples) and concatenated with x to form a $n+1$ vector X'. It is fed to the model which uses batch norms. The classification

of x (at index n of the result vector) is returned.

$$\begin{aligned} M(x;\theta) =& M_{BN(X')}(X';\theta)[n], \ \textit{with} \\ & X = \{x_1, ...x_n\}, \ x_i \sim \mathcal{X}, \\ & X' = X \diamond \{x\} \end{aligned} \tag{6}$$

VI. Evaluation

We evaluate our technique on two popular adversarial training methods, PGD [23] and TRADES [38], and show that norm shaping can substantially improve their effectiveness by achieving model accuracy comparable to well trained normal models and much better robustness. We also show our models' robustness under the strongest adaptive attack in which the adversary knows the set of norm shaping samples used in each query. We conduct an ablation study regarding the ratio between adversarial and clean samples. Our implementation is based on the standard PGD training code downloaded from [1]. All the experiments are run on a server equipped with two Intel Xeon Silver 4214 2.20GHz 12-core processors, 188 GB of RAM, and eight NVIDIA GeForce RTX 2080 Ti GPU cards.

Experiment Settings. We use CIFAR10 as our dataset. We use the same ResNet w32-10 structure (from the PGD code base) for all the models in our experiment to preclude differences induced by model structure.

TRADES. Besides PGD, we also use TRADES as a baseline. It is another popular adversarial training technique that co-chains clean and adversarial samples. For a given batch of clean samples, it generates adversarial samples that can maximize the statistical divergence of their classification results, using the Kullback–Leibler (KL) divergence metric [17]. The training mixes the benign samples and the corresponding adversarial samples, and updates model weights to minimize the cross-entropy loss of clean samples and the KL divergence between the two kinds of samples. TRADES's implementation is based on PyTorch whereas PGD's is on Tensorflow. To simplify comparison, we reimplement TRADES inside the PGD training framework. We observe that using KL divergence in the training loss degrades model accuracy. We hence disable it. Note that KL divergence is still used in adversarial sample generation.

Training Setting: We mostly reuse the default training configuration from [1], with 80,000 training steps, 8/255 perturbation bound, step size of 2, and 10 attack steps (in adversarial sample generation). More details can be found in the `config.json` file from [1]. The only difference is that we use a batch size of 64 instead of 128.

We use three training modes. The first is *adv. only*, meaning that only adversarial samples (not any clean samples) are used in training, which is the default setting of PGD. The second is *shaping in training*, in which norm shaping is only used in training but not in adversarial sample generation. In other words, the generation is following Eq. 4 such that the entire batch is perturbed. The third one is *shaping in both attack and training*. The ratio of clean and adversarial samples is always 3:1.

Attacks: We use five attacks to test the robustness of the trained models, PGD, C&W L2 attack [7], FGSM [13], Deepfool [22], and our norm shaping attack. The first four are from the Cleverhans framework [25], mostly using their default attack settings. We change the perturbation bound of FGSM from the pixel value of 8 to 16, as the original attack was too weak. The batch size is always 100. Given a batch of test samples, the attack code is run to generate adversarial samples, which are fed to the models for testing. Note that in the first four attacks, PNs are used in adversarial sample generation, which is their default setting. In the norm shaping attack (which uses BNs), the ratio of the unchanged (test) samples and the attacked samples are 3:1, the same as that in training.

Inference Setting: We have three inference settings. The first one is using the BNs of test samples (regardless clean or adversarial samples). The second is using the PNs, which is equivalent to the default setting of most existing works. In the third one, we mix a test sample with nine randomly selected clean training samples (using the random seeds provided in the configuration file of PGD) for norm shaping. We discard the classification results of the norm shaping samples, and report only the results of test samples. To evaluate model accuracy, test samples are those from the test set. To evaluate model robustness, test samples are those generated by the attacks (from clean test samples). They are called *BN*, *PN*, and *BN+Shaping* inference modes, respectively.

Main Results. Table I presents the results for PGD based adversarial trainings. The first column denotes the three training modes; the second column the attacks; columns 3-4, 5-6, 7-8 denote the clean and robust accuracies for the three inference modes. Specifically, the results in blue correspond to the default PGD, whereas those in red correspond to our method. The others are for the different possible configurations. We have the following observations. *First*, compared to the default PGD, our method achieves much better clean accuracy (0.942 versus 0.869) and almost consistently better robustness. For example, under the PGD attack, our method achieves 0.816 robust accuracy whereas the default PGD training can only achieve 0.47, which is consistent with the literature. *Second*, our method achieves close-to-the-best robustness for almost all attacks except our own shaping attack. Other settings such as using the adv-only and shaping-in-training modes have better robustness against our attack. The reason is that our attack is designed particularly against norm shaping, namely, perturbations are generated with norm shaping. Such attack may not be strong when shaping is not employed. *Third*, the overall best results are achieved when shaping is used in adversarial attack, training, and inference. Other settings, such as shaping only in training (the second big row) and using only BNs in inference (columns 3-4), achieve inferior results in both accuracy and robustness.

Table II presents the results for TRADES based adversarial trainings. We have similar observations. This illustrates that our method is effective regardless the underlying adversarial training methods. It also outperforms using a simple mix of

Training	Attack	BN		PN		BN+Shaping	
		Acc	R. Acc	Acc	R. Acc	Acc	R. Acc
Adv. only	PGD	0.877	0.501	0.869	0.47	0.873	0.504
	CW		0.865		0.821		0.864
	FGSM		0.391		0.376		0.395
	Deepfool		0.503		0.071		0.515
	Shaping		0.606		0.614		0.600
Shaping in Training	PGD	0.932	0.142	0.922	0.0	0.932	0.384
	CW		0.92		0.0		0.896
	FGSM		0.263		0.502		0.544
	Deepfool		0.929		0.051		0.903
	Shaping		0.696		0.727		0.447
Shaping in Attack+ Training	PGD	0.941	0.281	0.944	0.030	0.942	0.816
	CW		0.905		0.069		0.904
	FGSM		0.552		0.597		0.740
	Deepfool		0.914		0.037		0.911
	Shaping		0.481		0.494		0.533

TABLE I: Results for PGD based adversarial trainings

Training	Attack	BN		PN		BN+Shaping	
		Acc	R. Acc	Acc	R. Acc	Acc	R. Acc
Adv. only	PGD	0.890	0.476	0.888	0.462	0.889	0.481
	CW		0.880		0.845		0.880
	FGSM		0.351		0.347		0.359
	Deepfool		0.521		0.066		0.521
	Shaping		0.522		0.533		0.539
Shaping in Training	PGD	0.937	0.298	0.925	0.028	0.938	0.529
	CW		0.933		0.053		0.925
	FGSM		0.348		0.467		0.489
	Deepfool		0.941		0.052		0.924
	Shaping		0.729		0.851		0.472
Shaping in Attack+ Training	PGD	0.945	0.202	0.945	0.004	0.944	0.817
	CW		0.899		0.026		0.899
	FGSM		0.510		0.546		0.737
	Deepfool		0.908		0.037		0.914
	Shaping		0.499		0.521		0.542

TABLE II: Results for TRADES based adversarial trainings

clean and adversarial samples like in TRADES.

Our technique does not entail additional overhead compared to vanilla PGD or TRADES, its test time overhead is also small as that is only due to using batch and normalization.

Adaptive Attack. Although our threat model assumes the attacker does not know the norm shaping samples, we conduct an experiment to study the effectiveness when a stronger threat model is assumed in which the attacker knows the norm shaping samples such that he generates attack samples using the same BNs by shaping. This constitutes the strongest attack to our method. In Table III, the second and third columns show we use the models trained by PGD and TRADES with norm shaping. The last row shows the results. Observe that our models can still have over 0.50 robustness. In addition, we study a slightly weaker threat model, in which the attacker does not know the exact norm samples but he can access other training samples (and use them to perform norm shaping during attack). In this case, our robustness is slight better.

Ablation Study. In this study, we use different norm shaping ratios in training and inference and study our method's effectiveness. In training, we use the following ratios between adversarial and clean samples, 1:1, 1:2, 1:3, and 1:7, as

	PGD+Shaping	TRADES+Shaping
Slightly weaker attack	0.518	0.514
Adaptive attack	0.508	0.505

TABLE III: Adaptive attack results

	adv:clean in inference					
in training	**1:0**	**1:1**	**1:3**	**1:9**	**1:19**	**1:49**
1:1	0.477	0.478	0.476	0.476	0.477	0.476
1:2	0.385	0.837	0.835	0.753	0.668	0.573
1:3	0.281	0.692	0.820	0.817	0.813	0.815
1:7	0.039	0.138	0.706	0.781	0.776	0.779

TABLE IV: Impact of the ratio between adversarial and clean samples on robustness against PGD attack

shown by the first column in Table IV. In inference, we use 6 ratios as shown in the first row of the table. Ratio 1:0 means no norm shaping. The list of ratios in training and inference differ due to the divisibility of batch size (64 in training and 100 in inference). Here, we use the PGD attack from CleverHans and report the robust accuracy. Observe that when the training ratio is 1:1 (row 2), the different inference ratios make no difference. The robustness is consistently low, similar to the default PGD training. The reason is that since the clean samples are not dominating, the BNs follow a complex distribution as the BNs of the two kinds of samples are quite different. The training ratio 1:2 achieves the best robustness when the inference ratio are 1:1 and 1:3. However, its performance degrades when more clean samples are used in shaping. Its model accuracy is 0.92 (worse than 0.944 in our chosen setting). The training ratio 1:3 (the default setting) delivers the best overall performance. Also observe that with that training setting, the inference-time shaping needs to have at least the ratio of 1:3. Increasing the number of clean samples beyond this ratio has no benefit, but does not degrade performance either. It seems to indicate the BNs are sufficiently stable. When the training ratio is 1:7, the inference ratio ought to be 1:9 or beyond to get good results.

VII. Conclusion

We develop a novel adversarial training method that addresses the confoundings caused by batch normalization. We observe that batch normalization can undesirably degrade model accuracy, reduce attack strength, and cause a running target effect, making adversarial training fail to patch model weakness. We propose a norm shaping technique that forces the model to use batch norms all the time and ensures stability of batch norms by enforcing a fixed and large portion of clean samples in each batch. This ensures that the batch norms have a distribution similar to that of clean samples, suppressing the confoundings. Our experiments show that it can substantially improve existing adversarial training methods such as PGD and TRADES, achieving 0.94 model accuracy (compared to 0.88 by baselines) and 0.81 robustness against the PGD attack (compared to 0.47 by baselines).

Acknowledgement

This research was supported by IARPA TrojAI W911NF-19-S-0012. Any opinions, findings, and conclusions in this paper are those of the authors only and do not necessarily reflect the views of our sponsors.

References

[1] Pgd on cifar10 challenge. https://github.com/MadryLab/cifar10_challenge. 6

[2] Jean-Baptiste Alayrac, Jonathan Uesato, Po-Sen Huang, Alhussein Fawzi, Robert Stanforth, and Pushmeet Kohli. Are labels required for improving adversarial robustness? In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 3

[3] Mislav Balunovic and Martin Vechev. Adversarial training and provable defenses: Bridging the gap. In *International Conference on Learning Representations*, 2019. 1

[4] Philipp Benz, Chaoning Zhang, Adil Karjauv, and In So Kweon. Revisiting batch normalization for improving corruption robustness. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 494–503, January 2021. 1, 3

[5] Philipp Benz, Chaoning Zhang, and In So Kweon. Batch normalization increases adversarial vulnerability and decreases adversarial transferability: A non-robust feature perspective, 2020. 3

[6] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, AISec '17, page 3–14, New York, NY, USA, 2017. Association for Computing Machinery. 1

[7] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. In *IEEE Symposium on Security and Privacy (S&P)*, pages 39–57, 2017. 2, 6

[8] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang. Unlabeled data improves adversarial robustness. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 3

[9] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pages 1310–1320. PMLR, 2019. 1

[10] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017. 1

[11] Angus Galloway, Anna Golubeva, Thomas Tanay, Medhat Moussa, and Graham W. Taylor. Batch normalization is a cause of adversarial vulnerability, 2020. 3

[12] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, 2018. 1

[13] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. 1, 2, 3, 4, 6

[14] Tianyu Han, Sven Nebelung, Federico Pedersoli, Markus Zimmermann, Maximilian Schulze-Hagen, Michael Ho, Christoph Haarburger, Fabian Kiessling, Christiane Kuhl, Volkmar Schulz, et al. Advancing diagnostic performance and clinical usability of neural networks via adversarial training and dual batch normalization. *Nature communications*, 12(1):1–11, 2021. 3

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015. 1, 3

[16] Jinyuan Jia, Xiaoyu Cao, Binghui Wang, and Neil Zhenqiang Gong. Certified robustness for top-k predictions against adversarial perturbations via randomized smoothing. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. 1

[17] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951. 6

[18] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. 1, 3

[19] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 656–672. IEEE, 2019. 1

[20] Aishan Liu, Shiyu Tang, Xianglong Liu, Xinyun Chen, Lei Huang, Haotong Qin, Dawn Song, and Dacheng Tao. Towards defending multiple \ell_p-norm bounded adversarial perturbations via gated batch normalization. 2021. 3

[21] Shiqing Ma, Yingqi Liu, Guanhong Tao, W. Lee, and X. Zhang. Nic: Detecting adversarial samples with neural network invariant checking. In *NDSS*, 2019. 1

[22] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016. 2, 6

[23] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. 1, 2, 3, 6

[24] Tianyu Pang, Chao Du, Yinpeng Dong, and Jun Zhu. Towards robust detection of adversarial examples. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31. Curran Associates, Inc., 2018. 1

[25] Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Cihang Xie, Yash Sharma, Tom Brown, Aurko Roy, Alexander Matyasko, Vahid Behzadan, Karen Hambardzumyan, Zhishuai Zhang, Yi-Lin Juang, Zhi Li, Ryan Sheatsley, Abhibhav Garg, Jonathan Uesato, Willi Gierke, Yinpeng Dong, David Berthelot, Paul Hendricks, Jonas Rauber, and Rujun Long. Technical report on the cleverhans v2.1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*, 2018. 6

[26] Leslie Rice, Eric Wong, and J. Zico Kolter. Overfitting in adversarially robust deep learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 8093–8104. PMLR, 2020. 1, 3

[27] Chawin Sitawarin, Arvind Sridhar, and David Wagner. Improving the accuracy-robustness trade-off for dual-domain adversarial training. In *UDL*, 2021. 3

[28] A Sridhar, Chawin Sitawarin, and David Wagner. Mitigating adversarial training instability with batch normalization. In *Proceedings of International Conference on Learning Representation Workshop on Security and Safety in Machine Learning Systems*, 2021. 1, 2, 3

[29] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*. 1, 3

[30] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017. 1, 3

[31] Haotao Wang, Aston Zhang, Shuai Zheng, Xingjian Shi, Mu Li, and Zhangyang Wang. Removing batch normalization boosts adversarial training. In *International Conference on Machine Learning*, pages 23433–23445. PMLR, 2022. 3

[32] Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*, 2019. 1, 3

[33] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5286–5295. PMLR, 2018. 1

[34] Yuxin Wu and Justin Johnson. Rethinking "batch" in batchnorm, 2021. 2

[35] Cihang Xie and Alan Loddon Yuille. Intriguing properties of adversarial training at scale. *arXiv: Computer Vision and Pattern Recognition*, 2020. 1, 3

[36] Xuwang Yin, Soheil Kolouri, and Gustavo K Rohde. Gat: Generative adversarial training for adversarial example detection and robust classification. In *International Conference on Learning Representations (ICLR)*, 2019. 1

[37] Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. In *International Conference on Learning Representations*, 2020. 1

[38] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 7472–7482. PMLR, 2019. 1, 2, 3, 6