Splitting the Difference on Adversarial Training

Matan Levi  
Ben-Gurion University of the Negev  
matanle@post.bgu.ac.il

Aryeh Kontorovich  
Ben-Gurion University of the Negev  
karyeh@bgu.ac.il

Abstract
The existence of adversarial examples points to a basic weakness of deep neural networks. One of the most effective defenses against such examples, adversarial training, entails training models with some degree of robustness, usually at the expense of a degraded natural accuracy. Most adversarial training methods aim to learn a model that finds, for each class, a common decision boundary encompassing both the clean and perturbed examples. In this work, we take a fundamentally different approach by treating the perturbed examples of each class as a separate class to be learned, effectively splitting each class into two classes: “clean” and “adversarial.” This split doubles the number of classes to be learned, but at the same time considerably simplifies the decision boundaries. We provide a theoretical plausibility argument that sheds some light on the conditions under which our approach can be expected to be beneficial. Likewise, we empirically demonstrate that our method learns robust models while attaining optimal or near-optimal natural accuracy, e.g., on CIFAR-10 we obtain near-optimal natural accuracy of 95.01% alongside significant robustness across multiple tasks. The ability to achieve such near-optimal natural accuracy, while maintaining a significant level of robustness, makes our method applicable to real-world applications where natural accuracy is at a premium. As a whole, our main contribution is a general method that confers a significant level of robustness upon classifiers with only minor or negligible degradation of their natural accuracy.

1 Introduction
Despite their success in a wide variety of challenging tasks, Neural Networks are brittle when faced with small, imperceptible perturbations to their input; these are commonly referred to as adversarial examples, which will, with high probability, alter the neural network’s classification [10, 11, 25, 29, 41, 47, 59, 68, 69, 72, 88]. Early methods for defense against such attacks were soon broken by stronger adversaries [4]; subsequently, adversarial training emerged as one of the most effective defenses [29, 45, 68, 96]. These adversarial training techniques aim to learn robust models by solving a min-max optimization problem.

While the inner maximization searches for worst-case adversarial examples during training, and then augments the training data with them, the outer minimization optimizes across model parameters given natural and adv. examples.

Usually, in standard adversarial training methods, each generated adversarial example is annotated with the source class label. Some works also attach a domain label (clean or adversarial) for enhanced techniques, such as using an additional domain classifier [44], adversarial examples detection [10, 70], etc. On the contrary, we hypothesize that adversarial examples generated from a given source class induce a totally distinct class distribution. Therefore, in case one wishes to avoid significant natural accuracy degradation, adversarial training should be adjusted to take these additional classes into account during training. Overall, we make the following contributions:

• We introduce a novel approach for training robust models, which departs from the established paradigm of attempting to learn a common decision boundary for each natural class and its adversarially perturbed version. Rather, we claim that for each class, the adversarial perturbations induce a distinct distribution on the examples, so much so that it makes more sense to learn it as a separate label, rather than attempting to shoehorn it into the original one. Thus, our method doubles the number of classes but ends up learning much simpler decision boundaries; we provide both theory and experiments in evidence of the efficacy of this trade-off (more below). To our knowledge, this approach of “splitting the difference” (which we formally dub Double boundary adversarial training, DBAT), is completely novel in the adversarial training setting.

• We perform a comprehensive battery of experiments to demonstrate that our approach learns robust models
while also achieving the highest reported natural accuracy, with a significant margin across different datasets. This optimal or near-optimal natural accuracy makes our method applicable for real-world applications (autonomous vehicles, face recognition systems, healthcare monitoring, and diagnosis, etc.) that cannot sacrifice natural accuracy for robustness. We stress that our aim is not to compete with the state-of-the-art models on robustness, but rather a general-purpose technique for endowing a classifier with a significant level of robustness, while only incurring a minor degradation of natural accuracy. 

In §4, we provide a rudimentary plausibility argument to shed some theoretical light on the statistical trade-off presented by DBAT: an increased number of classes to learn, but with much simpler boundaries.

2 Related work

Since the discovery of adversarial examples by [68], a wide range of defenses were proposed to enhance robustness. Among these, adversarial training [29,45] emerged as one of the most successful methods to train robust models. Madry et al. [45] proposed a technique, commonly referred to as standard Adversarial Training (AT), to minimize the cross entropy loss only on adversarial examples with respect to the original class labels. Throughout the years, standard adversarial training was enhanced in various ways [7] – with changes in the regularization terms [28,38,40,42,44,80,96], model ensemble [50,72,90], adversarial training with adaptive attack budget [15,23], curriculum adversarial training [9,79,97], utilizing out-of-distribution data [43], applying Stochastic Weight Averaging (SWA) [36] to flatten the adversarial loss landscape [14,31], adapting adversarial training to model weights using Adversarial Weight Perturbation (AWP) [73,85], and combining adversarial training with data augmentation techniques [32,56,57] and synthetically generated data [49,61,81,89].

Other lines of research include theoretically certified approaches [16,30,53,54,63,82,84], computationally efficient adversarial training [2,62,67,83,94], robust overfitting and possible mitigations [58], semi/un-supervised adversarial training [12,75,93], adversarial self-training and pretraining [13,37], incorporating domain adaptation alongside adversarial training [44,66], and robust model architecture and custom building blocks [77,86,87,95]. Specifically, normalizer-free robust training (NoFrost) [77] suggested removing all batch normalization (BN) layers from the network during AT, but this approach was shown to have a negative effect on the robustness against stronger attacks. 

Some well-known methods include [96], who proposed the method TRADES, which uses the Kullback-Leibler (KL) divergence as a regularization term to push the decision boundary away from the data.

Most related methods to DBAT are ones that suggest changes to the regularization terms of adversarial training (AT) with the goal of reducing natural accuracy degradation in AT. A recent work by [20], named LBGAT, used an additional Mean Square Error (MSE) regularization term between the logits of a natural model, alongside the robust. In [15], authors suggested a work named Customized Adversarial Training (CAT) which adaptively customizes the perturbation level and the corresponding label for each training sample, but was later shown to suffer from obfuscated gradients [64]. Another work [52] suggested Helper Adversarial Training (HAT), which attempts to mimic the discriminative features learned by standard trained networks to improve the accuracy of clean samples with the goal of improving natural accuracy. Recently, Universal Inverse Adversarial Training (UIAT) was suggested by [24] to encourage the model to produce similar output probabilities for an adversarial example and its “in-
verse adversarial” counterpart, where the counterpart is generated by maximizing the likelihood in the neighborhood of the natural example. Additionally, the authors of [78] recently suggested Generalist, which consists of two base learners separately trained within their respective fields and a global learner that aggregates the parameters of base learners during the training process. The parameters of base learners are collected and combined to form a global learner at intervals during the training process.

In contrast to all of the aforementioned methods, our work suggests a fundamentally different approach. While other methods treated the generated adversarial examples of a given class as additional instances of that same class when learning the class boundaries (or even used only the adversarial examples), our method acknowledges the fact that adversarial examples induce additional class distributions on the source dataset, which essentially doubles the number of classes in the dataset, and therefore these examples should be treated as additional classes of the dataset. We underline that our goal is not to improve robust accuracy compared to the current state-of-the-art in robustness, but rather to equip models with a significant level of robustness, while keeping their natural accuracy as high as possible.

3 Double Boundary Adversarial Training

In this section, we introduce our approach for training robust models, Double Boundary Adversarial Training, DBAT. A conceptual illustration is presented in Figure 1, and is supported empirically in Appendix D, where we test if such a case is possible in real-world scenarios by calculating the distance histogram of random examples to the decision boundary.

3.1 Motivation behind Double Boundary Adversarial Training

Tsipras et al. [74] argued that robustness may be at odds with natural accuracy, and usually the trade-off is inherent. We concur that this indeed is typically the case when adversarial examples are assigned to the same class as the natural examples they were generated from. However, when separating the adversarial examples from their source, and generating new parallel adversarial classes, we may be able to maintain natural accuracy while still achieving significant robustness.

Therefore, we suggest an alternative approach to learning robust models. Our main hypothesis is that natural examples and their adversarial counterparts should not necessarily be assigned to the same class. Instead, for each class, we learn an additional counterpart adversarial class, which will be assigned to the adversarial examples. In essence, the number of classes in the dataset is doubled.

In other words, instead of learning shared boundaries for both natural examples and their adversarial counterparts, where adversarial examples are expected to reside in the same class as their natural counterparts, we suggest treating adversarial examples as additional classes in the dataset.

While other methods can be thought of as modifying existing boundaries, DBAT learns boundaries for completely new dynamically generated adversarial classes. We hypothesize that this behavior creates a trade-off, where on one hand, DBAT does not induce significant changes to existing boundaries for natural classes in terms of complexity, and keeps them smoother (as we empirically demonstrate on the synthetic dataset experiment in §3.4 and Figure 2) - which mitigates the drop in natural accuracy. But on the other hand - it’s a more challenging task to learn completely new classes, which in turn can impact robustness in some tasks.

We highlight that our aim is not to compete with the state-of-the-art on robustness, but rather to find a general-purpose technique that equips classifiers with a significant level of robustness with only minor or neglectable degradation of their natural accuracy.

3.2 Training procedure

During the training process, our goal is to learn additional classes, one for each in the original class set. Given a dataset \( S = \{ (x_i, y_i) \}_{i=1}^n \) with \( C \) classes \( Y = \{ 0, 1, ..., C - 1 \} \), we define a new class space \( Y_{DBAT} = \{ 0, 1, ..., C - 1, C + 1, ..., C + k \} \) where class label \( C + k, k \geq 0 \) is the label of the adversarial examples generated for class \( k \).

For each natural example \((x_i, y_i)\), DBAT generates an adversarial example \( x_i' \) using targeted-PGD with a random target. Then, the adversarial example is assigned with the adversarial class \( y_i + C \) which corresponds to the natural class \( y_i \) of the natural example. To summarize, for each natural example \((x_i, y_i)\), we generate adversarial example and assign it the corresponding adversarial class, \((x_i', y_i + C)\). Algorithm 1 describes the training procedure.

Remark. We note that targeted-PGD typically does not outperform untargeted-PGD when used with standard adversarial training methods. That said, we argue that this observation is not applicable to our setting. First, since we aim to learn new generic adversarial classes, it stands to reason that class diversity will be conducive to generalization. Therefore, using random targeted-PGD mitigates the scenario where adversarial examples generated by untargeted attacks for a given class focus on small/specific regions of the manifold. Additionally, using untargeted PGD, attacks can potentially be directed to the adversarial class corresponding to the natural one. To avoid the later, one can use untargeted PGD only on the original classes, by adding a projection back to the original classes during the optimization. Finally, we also experimented with using Least-Likely targeted-PGD. See Appendix B for results comparison.
3.3 Inference procedure

At inference time, the model will output a probability vector \( v \) of size \( |v| = 2 \cdot C \) which corresponds to the double number of classes used during training. However, the dataset originally has only \( C \) classes. Therefore, as our final class prediction, we use the following formula:

\[
\hat{c} = \arg\max_{0 \leq c \leq C} v_c^\ell, \quad (1)
\]

In other words, the final class prediction is taken as the class with the maximum probability. If this class is one of the adversarial classes, we return to its natural counterpart.

3.4 Illustrating DBAT’s Decision Boundaries using a Synthetic Dataset

In Figure 2 we illustrate how DBAT can learn simpler and smoother decision boundaries by applying it to a synthetic dataset. We exhibit the decision boundaries for standard AT and DBAT. The dataset is composed of isotropic Gaussian blobs with a cluster standard deviation of 0.1 and two features generated using the make_blobs from [51]. The number of samples for each blob is 10,000. The adversary was given a budget of \( \epsilon = 1.2 \) optimized for six steps with a step size of 0.2. This enables some of the samples to cross the decision boundary. As can be seen in Figure 2c, our method learns much smoother and simpler decision boundaries as compared to standard adversarial training in Figure 2b.

4 Theoretical analysis

We provide a theoretical plausibility argument for the empirical success of our approach, in the following somewhat idealized setting. We identify a phenomenon, which we term the DBAT advantage, which, when applicable, justifies the use of our technique.

Here we assume familiarity with the basic notions of PAC learning, such as the sample error of a hypothesis, \( \hat{e}(h) \), its generalization error, \( err(h) \), and the Vapnik-Chervonenkis (VC) dimension of a concept class; these may all be found, e.g., in [3]. Suppose that one trains a \( k \)-multiclass classifier by reducing it to \( k \) binary classification problems via the standard 1-vs-all method (i.e., a separate in-class/out-class binary classifier is trained for each of the \( k \) classes). Suppose for simplicity that each of the classifiers is trained using the same concept class \( H \) of VC-dimension \( V \). If \( h_i \) is the classifier trained for the \( i \)-th class on a sample of size \( n \) with sample error \( \hat{e}(h_i) \), then the agnostic PAC bound [3, Theorem 4.9] implies that with probability at least 1 - \( \delta \),

\[
err(h_i) - \hat{e}(h_i) \leq c \left( \sqrt{\frac{V + \log(1/\delta)}{n}} \right), \quad (3)
\]

where \( err(h_i) \) is the generalization error and \( c > 0 \) is a universal constant.

Claim. The following form of (3) holds for all of the \( k \) classifiers, with probability at least 1 - \( \delta \), simultaneously:

\[
\max_{1 \leq i \leq k} \hat{e}(h_i) \leq c \left( \sqrt{\frac{V + \log(1/\delta)}{n}} \right) \quad (4)
\]

Proof. One sets \( \delta' = \delta/k \), which guarantees, with probability at least 1 - \( \delta' \), a generalization error of at most \( c \sqrt{\frac{V + \log(1/\delta')}{n}} \) for each class individually, and hence, by a union bound, for all classes simultaneously, with probability of at least 1 - \( k\delta' = 1 - \delta \).

But now suppose that we can express each \( h \in H \) as a union of two simpler concepts: \( h = h_1 \cup h_2 \), where \( h_1, h_2 \in H' \), and the latter has VC-dimension, say, \( V/2 \). In this case, we can formulate the learning problem as a 2\( k \)-multiclass classification problem, over the concept class \( H' \). By assigning \( V/2 \) and \( 2k \), the corresponding bound in (4) will now behave as:

\[
\sqrt{\frac{(V/2 + \log(2k/\delta))n}{n}}\quad (5)
\]

— which, for constant \( \delta \) and large \( V \), constitutes considerable savings in sample complexity. The improvement in sample complexity will be even more significant as we consider \( \ell \)-fold (rather than just 2-fold) unions of basic concepts: \( h = h_1 \cup h_2 \cup \ldots \cup h_{\ell} \), \( h_i \in H' \). We will refer to this phenomenon — in which decreasing hypothesis complexity while increasing the number of classes reduces the overall sample complexity — as the DBAT advantage, and discuss it in greater detail below.
We will illustrate this phenomenon in some detail on the natural example of halfspaces and Euclidean balls in $\mathbb{R}^d$. Since providing the requisite background on Vapnik-Chervonenkis (VC) theory (in particular: shattering, VC-dimension) is beyond the scope of the paper, we refer the reader to [46].

**Halfspaces.**

**Claim 1.** If $H'$ is the collection of all homogeneous (going through the origin) halfspaces $\mathbb{R}^d$, then the VC-dimension of $H'$ is $d + 1$.

**Proof.** It is shown in [46, Example 3.2] that the VC-dimension of general halfspaces in $\mathbb{R}^d$ is $d + 1$. The restriction that the halfspace contain the origin can be ensured by translating any shattered set $\{x_1, \ldots, x_{d+1}\}$ by $x_1$ to obtain the shattered set $\{x_2 - x_1, \ldots, x_{d+1} - x_1\}$ of size $d$. This shows that homogeneous hyperplanes have VC-dimension 1 less than the general ones, i.e., $d$.

**Claim 2.** For $H'$ as above (the collection of all homogeneous halfspaces in $\mathbb{R}^d$), the set of all 2-fold unions of concepts from $H'$ will have VC-dimension at least twice that of $H$.

**Proof.** For the lower bound, it suffices to find $d$ points in the positive orthant shattered by a set of $2^d$ homogeneous halfspaces $H_1 \subset H'$, as well as another set of $d$ points in the negative orthant shattered by another set of $2^d$ homogeneous halfspaces $H_2 \subset H'$, such that each $h \in H_1$ labels the negative orthant positive, while each $h \in H_2$ labels the positive orthant positive. Evidently, the set of pairwise unions of $h_1 \in H_1$ and $h_2 \in H_2$ shatters the combined set of $2d$ points.

This example illustrates that 2-fold unions of simple classifiers can double the VC dimension of the hypothesis class. In the more general case of $\ell$-fold unions of hyperplanes, it is known [19] that the VC-dimension is $\Omega(\ell d \log \ell)$, so the increase in sample complexity is even more significant. Moreover, [19] showed that this continues to be true for many other kinds of Boolean aggregations: intersections, XORs etc.

**Euclidean balls.**

**Claim 3.** The VC-dimension of Euclidean balls in $\mathbb{R}^d$ is $d + 1$.

**Proof.** This is a well-known fact, which we prove for completeness and also because the argument will be useful in the sequel.

Both the upper and lower bounds on the VC-dimension of balls rely on the fact that locally, these act like halfspaces: any two finite sets separated by a halfspace can also be separated by a ball of large enough radius (see Figure 4).

This argument is enough to establish the lower bound: any set that is shattered by general halfspaces is also shattered by Euclidean balls (see Figure 5), and we know from [46, Example 3.2] that such a set can be as large as $d + 1$.

For the upper bound, we invoke Radon’s theorem [46, Theorem 3.4]: Any set $S$ of $d + 2$ points in $\mathbb{R}^d$ can be partitioned into two subsets $S_1$ and $S_2$ such that the convex hulls of $S_1$ and $S_2$ intersect. Such a partition will be called a *Radon partition*. Suppose, for a contradiction, that the Euclidean balls shatter some set $S$ of $d + 2$ points. Then there exists a Radon partition of these into $S_1$ and $S_2$. But shattering means that some ball $B_1$ contains $S_1$ and not $S_2$, while another ball $B_2$ contains $S_2$ but not $S_1$.

This means that $S_1$ and $S_2$ must be separable by a hyperplane. We conclude that the Euclidean balls cannot shatter any more points that the halfspaces, which is at most $d + 1$.

**Claim 4.** If $H'$ is the collection of the Euclidean balls in $\mathbb{R}^d$ (with VC-dimension $d + 1$, as shown above), and $H$ is the set of all 2-fold unions of concepts from $H'$, then the VC-dimension of $H$ is at least $2d$.

**Proof.** The argument proceeds by the same reduction from balls to halfspaces employed in the proof of the lower bound in Claim 3: any set that can be shattered by halfspaces can also be shattered by balls. Now, as in the proof of Claim 2, we construct two disjoint sets shattered by homogeneous halfspaces, consisting of $d$ points each, in the positive and negative orthants, respectively. Each is also shattered by balls, and by dilating the points sufficiently far from the origin, we can ensure that the $2^d$ balls shattering the positive-orthant set are...
Figure 3: Visualizing the logits of DBAT on CIFAR-10 test set using T-SNE [76] with two components on the model output for (a) natural examples (b) adversarial examples with their new generated adversarial classes (c) combined 2-D visualization of both natural and adversarial examples with all 20 classes (same color for natural class and its adversarial counterpart). (d) same plot as in c, but colored in two colors to observe the separation between natural and adversarial examples. We can observe the strong separation between classes obtained by DBAT, for both original and newly generated classes. Interestingly, adversarial and natural examples almost don’t mix, and the majority of mismatches within each cluster are from the same domain (adversarial/natural).

Figure 4: Two finite sets separated by a halfspace can also be separated by a ball of large enough radius.

Figure 5: A finite set that is shattered by balls is also shattered by halfspaces.

disjoint from their counterparts shattering the negative-orthant set. Thus, the 2-fold unions of Euclidean balls shatter a set of size $2d$.

The above discussion was more of a proof-of-concept illustration, since VC-dimension is not a particularly practical tool in analyzing deep neural networks with a large number of weights. In Appendix C, we show that the thrust of our point continues to hold for the Rademacher complexity as well, which is far more practical as far as providing finite-sample generalization bounds [8, 91]. Using the analysis of [26], we show that the Rademacher complexity of $\ell$-fold unions grows with $\ell$ roughly as $\sqrt{\ell}$.

An additional qualification of our plausibility argument is that adversarial loss is distinct from the 0-1 loss discussed above. This is indeed a limitation of our analysis, although in some instances it is possible to control adversarial risk via a VC-type analysis [5, Theorem 2]. Finally, an implicit assumption we have made above is that the adversarial perturbations are non-adaptive: the adversary has fixed a (possibly, stochas-
tic) perturbation function in advance of seeing any data — e.g., a neural network trained on a hold-out set, similar to black-box settings. This lets us argue that the examples continue to be iid, under a new (unknown, perturbed) distribution. This assumption, while not entirely realistic, is often made to facilitate analysis [5].

Modulo these qualifications, the above discussion provides evidence that when training $k$ classifiers from a concept class with high complexity, it may be advantageous to decompose them into unions of simpler classifiers. The blow-up of the number of classifiers is more than compensated in the reduction of classifier complexity.

5 Experiments

To emphasize the advantage of Double Boundary Adversarial Training, we conduct extensive evaluations. The evaluation process of DBAT includes white-box and black-box settings, Auto-Attack, natural corruptions [35], unforeseen adversaries, and ablation studies. All results are averaged over 5 runs while omitting one standard deviation. These evaluations demonstrate that the results obtained are not a consequence of what is commonly referred to as obfuscated gradients [4].

We compare our method to some of the most well-known adversarial training methods – Standard AT [45], and TRADES [96], alongside related work – LBGAT [73], Generalist [78], CAT [15], HAT [52], and UIAT [24]. Our evaluation starts with the common CIFAR-10 benchmark. In §5.4 and §5.6, we demonstrate the generalization of our method to other datasets by experiments on CIFAR-100 [39] and SVHN [48]. We use the WRN-34-10 [92] architecture for CIFAR-10 and CIFAR-100, and the PreAct ResNet-18 for SVHN. As suggested in [57], we combine Stochastic Weight Averaging (SWA) [36], and Cutout [21] with window length eight. We used "concatenated batches" as suggested by [65]. Attacks are generated using $\ell_\infty$-PGD with $\varepsilon = 8/255$, and perturbation step size $1/255$ for 10 attack steps. Full experiment settings are detailed in Appendix A.

5.1 Threat model

Our trained model outputs a vector whose dimension is twice the number of classes in the dataset. That is, one-half of the coordinates corresponds to the original classes, while the second half corresponds to the new adversarial classes. Therefore, when considering the adversary’s capabilities, specifically for untargeted white-box attacks, we need to explicitly define how the optimization is done. Recall that the aggregation/projection function described in §3.3 takes place only at inference time. Moreover, since the projection is not part of the computation graph, the defender can switch it to any desired metric (max, mean, median, log, exp, etc.) at any time during inference, without updating the network. Therefore, there are three possible adversaries:

First, the most basic adversary is one who does not utilize any projection function while attacking.

Second, a more advanced (and perhaps most realistic) adversary, is one who knows that the defender is utilizing a projection function, but does not have inference time access to the defender (e.g., a model was published at model zoos), and needs to conjecture the projection function while attacking.

Third, and most powerful adversary (although the least realistic), is one who can access not only the entire network parameters but also real-time access to the defender’s system and projection function at any given time during inference.

Table 1: Natural, PGD, and Auto-Attack (AA) robust accuracy against the fully adaptive white-box perfect knowledge adversary (i.e., "Inference real-time access"). The attack is an $\ell_\infty$ attack on CIFAR-10 with WRN-34-10. Similar to our method, the presented results are for models that do not utilize additional data during training. The Natural method refers to a model trained using standard training under the same hyper-parameters settings. Green represents the best natural accuracy among the robust models. Orange represents the second-best natural accuracy among the robust models. The range alongside the green arrow represents the natural accuracy improvement compared to the other methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Natural Acc.</th>
<th>PGD</th>
<th>AA</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBAT (OURS)</td>
<td><strong>95.01</strong> (↑4–10.1%)</td>
<td>54.61</td>
<td>40.08</td>
</tr>
<tr>
<td>AT</td>
<td>85.10</td>
<td>54.46</td>
<td>51.52</td>
</tr>
<tr>
<td>TRADES</td>
<td>84.92</td>
<td>55.56</td>
<td>53.08</td>
</tr>
<tr>
<td>LBGAT</td>
<td>88.22</td>
<td>54.31</td>
<td>52.86</td>
</tr>
<tr>
<td>Generalist</td>
<td><strong>91.03</strong></td>
<td>56.92</td>
<td>52.91</td>
</tr>
<tr>
<td>HAT</td>
<td>84.86</td>
<td>52.30</td>
<td>48.85</td>
</tr>
<tr>
<td>UIAT</td>
<td>85.01</td>
<td>54.63</td>
<td>49.11</td>
</tr>
<tr>
<td>CAT</td>
<td>89.61</td>
<td>73.38</td>
<td>34.78</td>
</tr>
</tbody>
</table>

| Natural | 95.43 | 0 | 0 |

Throughout the paper, we compare against the most powerful adversary. In §5.8, we present additional experiments to demonstrate how different access to the inference time projection function affects the adversary’s strength (i.e., the attacker’s ability to degrade robust accuracy). As for black-box attacks analysis, we evaluate against two types of adversaries: naturally trained surrogate models, and other adversarially trained models. For natural corruptions, the corruptions are generated independently from the trained model.
5.2 White/Black-box and Auto-Attack Evaluation

White-box/Black-box PGD Robustness. We present DBAT’s $\ell_\infty$-PGD white-box and black-box results compared to a variety of adversarial training methods. Attacks are generated with $\epsilon = 8/255$, and perturbation step size $1/255$ and 10 attack steps. On CIFAR-10, DBAT’s results are in line with the SOTA methods under black-box attacks. For PGD white-box attacks, DBAT achieves significant PGD robustness (e.g., 54.25% with PGD$_{1000}$), similar to the other methods, with near-optimal natural accuracy of 95.01% (compared to 95.43% for a naturally trained model). Additionally, in Figure 3 we visually present the strong class separation obtained by DBAT for the original classes, the newly generated adversarial classes, and the combination of all the 20 classes for CIFAR-10.

Auto-Attack Evaluation. We evaluate DBAT on Auto-Attack, an ensemble of diverse attacks: APGD, APGD-DLR [18], Square [1], and FAB [17]. As described in Table 1, our method reaches near-optimal natural accuracy (compared to a naturally-trained model) while still maintaining significant robustness when tested against AA. We note that Auto-Attack results are not as good as PGD results. We ascribe the difference to the adversarial classes that were generated using $\ell_\infty$-PGD and are therefore oriented towards PGD adversaries. It can be empirically evidenced in the “unforeseen attacks” (Figure 6), where our results on attacks such as C&W are good, but our results on the different PGD adversaries with different norms ($\ell_\infty, \ell_2, \ell_1$) are better.

5.3 Unforeseen Adversaries Robustness

To further demonstrate that our method does not suffer from false robustness, we test it against different adversaries that were not observed during training, including $\ell_2$-PGD, $\ell_1$-PGD, $\ell_\infty$-DeepFool, and $\ell_2$-DeepFool [47] implemented by Foolbox [55], and CW$_\infty$ [11]. We applied white-box attacks, with common attack budgets of 12 for $\ell_1$-PGD, 0.5 for $\ell_2$-PGD, 0.02 overshoot for DeepFool, and 8/255 for CW$_\infty$. Results are visualized in Figure 6. Our method significantly improves results (except for CW$_\infty$), even on unforeseen adversaries. DBAT improves $\ell_2$-PGD by up to 14%, $\ell_1$-PGD by up to 20%, $\ell_2$-DeepFool by up to 10%, and $\ell_\infty$-DeepFool by up to 16%.

Feature Adversaries. We demonstrate the effectiveness of DBAT to so-called “adaptive adversaries”: those that try to circumvent our defense by ignoring the projection function during the optimization process [71], and show that they cannot evade our defense. To do so, we tested DBAT on CIFAR-10 against feature and logit-level adversaries: Kullback–Leibler divergence (KLD) attack on the probabilities vectors [96], $l_2$ logit-matching attack [71] on adversarial examples and their corresponding natural examples, and lastly, a feature adversary suggested in [60]. We used $\epsilon = 8/255$, $\delta = 1/255$, and ran for 500 iterations. Results are presented in Table 2. DBAT presents notably impressive and strong results against feature
and logit-level adversaries. We also tried attacking the inner layers (and combinations of layers), in addition to attacking the feature representation layer, but we noticed it did not improve the attack success rate. These results also support our claim in §5.1, that an adaptive attacker will benefit from using the projection function in the attack optimization.

Table 2: CIFAR-10 results against feature and logit level adversaries.

<table>
<thead>
<tr>
<th>Adversary</th>
<th>Robust Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>KLD</td>
<td>85.9</td>
</tr>
<tr>
<td>$l_2$ Logit Matching</td>
<td>84.5</td>
</tr>
<tr>
<td>Feature Adversary [60]</td>
<td>86.8</td>
</tr>
</tbody>
</table>

Figure 7: CIFAR-10C accuracy comparison results between different methods over all 18 natural corruption types, including noise, blur, weather, and digital categories.

Figure 8: CIFAR-100C accuracy comparison results between different methods over all 18 natural corruption types, including noise, blur, weather, and digital categories.

5.4 Natural Corruptions Robustness

We demonstrate the effectiveness of DBAT when facing natural corruptions, as proposed by [35]. This corruptions benchmark dataset consists of 18 diverse corruption types. It covers noise, blur, weather, and digital categories. As the researchers claimed, research that improves performance on this benchmark should indicate general robustness gains, as the corruptions are varied and great in number. These corruptions each have five different levels of severity. To test DBAT, we use the CIFAR-10-C and CIFAR-100C corruptions benchmarks. Note that the corruptions are model-independent. As demonstrated in Figures 7 and 8, our method outperforms the other methods by a significant margin on all corruption types.

5.5 Clean vs. Robust Accuracy Trade-off

Clean and robust accuracy trade-off. Originally, the clean and adversarial classes were equally weighted during training. Meaning, given that $\lambda$ is the weighting factor for the adversarial classes, we set $\lambda = 1$ in our experiments.

In the following experiment, we run an extensive evaluation to show how the trade-off between natural and robust accuracy changes as we weigh the loss on the natural and adversarial classes differently, i.e., how the natural and robust accuracy changes as we change the values of $\lambda$. We use CIFAR-10 with the same experiment settings described above. We report Auto-Attack (AA) results, as well as natural accuracy results. In Figure 9 and Table 3, we plot DBAT’s and TRADES’s trade-off between natural and Auto-Attack robust accuracy as the weighting factor, $\lambda$, varies the trade-off between the natural and adversarial classes. For DBAT, we compare against the fully adaptive white-box perfect knowledge adversary (i.e., "Inference real-time access"). Not surprisingly, as we increase $\lambda$, clean accuracy decreases while robust accuracy increases, and vice-versa. However, as can be seen, the changes in natural accuracy for DBAT are relatively small, even though $\lambda$
Figure 9: Natural and AutoAttack robust accuracy trade-off, for DBAT and TRADES on CIFAR-10, as we vary the hyper-parameter $\lambda$ that controls the weight we put on the natural and adversarial classes. The numbers on the graph represent the value of $\lambda$ for the specific trade-off.

was changed between a wide range of 0.1 and 8.

Table 3: DBAT’s natural and Auto-Attack accuracy trade-off on CIFAR-10 as the weighting factor $\lambda$ varies between 0.1 and 8 (where 1 is the default value used in the experiments).

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>Natural</th>
<th>Auto-Attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>96.81</td>
<td>18.40</td>
</tr>
<tr>
<td>0.2</td>
<td>96.58</td>
<td>20.50</td>
</tr>
<tr>
<td>0.5</td>
<td>95.70</td>
<td>34.56</td>
</tr>
<tr>
<td>1</td>
<td>95.01</td>
<td>40.08</td>
</tr>
<tr>
<td>2</td>
<td>93.52</td>
<td>42.97</td>
</tr>
<tr>
<td>4</td>
<td>93.24</td>
<td>44.47</td>
</tr>
<tr>
<td>6</td>
<td>92.76</td>
<td>44.88</td>
</tr>
<tr>
<td>8</td>
<td>92.56</td>
<td>45.91</td>
</tr>
</tbody>
</table>

We also noticed that when increasing $\lambda$ beyond a value of 10, the model started to diverge. We attribute this behavior to the fact that we are over-weighting dynamically newly generated classes, which in turn affects the model’s ability to converge. We noticed that a simple warm start of $\lambda$ can help to some extent. As for TRADES, we’ve noticed that as we decrease $\lambda$ below 0.1, TRADES was not able to learn robust models.

Overall, we’ve demonstrated how reducing the trade-off parameter $\lambda$, TRADES was not able to match DBAT’s clean accuracy without losing robust accuracy almost entirely. This is another empirical evidence of DBAT’s unique ability to learn models with optimal or near-optimal natural accuracy and a significant level of robustness.

5.6 Generalization Across Datasets

To further demonstrate that DBAT’s robust-natural trade-off is indeed good compared to other methods, we adopt the recently proposed metric, F1-robust, suggested by [44] which was specifically designed as a balanced measurement for robust and natural accuracy. Results are presented in Table 4.

Table 4: Results on CIFAR-10 using the recently proposed F1-robust metric specifically designed as a balanced measurement for robust and natural accuracy. Higher is better.

<table>
<thead>
<tr>
<th>Defense</th>
<th>F1-Robust</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBAT</td>
<td>0.710</td>
</tr>
<tr>
<td>AT</td>
<td>0.657</td>
</tr>
<tr>
<td>TRADES</td>
<td>0.659</td>
</tr>
<tr>
<td>LBGAT</td>
<td>0.670</td>
</tr>
<tr>
<td>Generalist</td>
<td>0.685</td>
</tr>
<tr>
<td>HAT</td>
<td>0.622</td>
</tr>
<tr>
<td>UIAT</td>
<td>0.645</td>
</tr>
</tbody>
</table>

SVHN evaluation. For SVHN, DBAT achieves an improvement of 3-5% under black-box attacks. When tested against white-box PGD attacks, DBAT achieves significant PGD robustness of 53.40% (PGD$_{20}$), similar to other popular AT methods. Additionally, DBAT reaches a natural accuracy of 96.86%, compared to 96.85% for a naturally trained model under the same architecture and settings. Meaning, DBAT exhibits no reduction in natural accuracy, while also achieving significant robust accuracy under various strong adversaries.

CIFAR-100 evaluation. For CIFAR-100, DBAT achieves an improvement of 6-12% under black-box attacks. When tested in PGD white-box attacks, DBAT still achieves significant PGD robustness, e.g., 29.95% with PGD$_{20}$, similar to other popular AT methods. We note that Auto-Attack robustness is lower than the other methods, possibly due to the greater diversity in the dataset and the small number of examples in each class, which makes it more difficult to learn new adversarial class boundaries. DBAT achieves significant Auto-Attack robustness. Moreover, DBAT reaches a natural accuracy of 75.18%, compared to 79.30% for a naturally trained model.
Table 5: Natural, PGD, and Auto-Attack (AA) robust accuracy against the fully adaptive white-box perfect knowledge adversary (i.e., "Inference real-time access"). The attack is an \( \ell_\infty \)-PGD attack on SVHN and CIFAR-100. Similar to our method, the presented results are for models that do not utilize additional data during training. The Natural method refers to a model trained using standard training under the same hyper-parameters settings. Green represents the best natural accuracy among the robust models. Orange represents the second-best natural accuracy among the robust models. The range alongside the green arrow represents the natural accuracy improvement compared to the other methods.

![Table 5](image)

trained model under the same settings. Compared to the other methods, DBAT improves natural accuracy by 14.5-18.5%.

5.7 Ablation Studies

DBAT core components. We demonstrate the performance gain obtained by our method by removing the two parts that are not at the core of DBAT – SWA and Cutout. We use the CIFAR-10 dataset, with WRN-34-10, and report the Auto-Attack (AA) results when removing SWA and Cutout. When removing SWA and Cutout we observe that their total contribution to DBAT is 2.21% in natural accuracy and 3.55% in robust accuracy. Additionally, in Table 7 we present the results using different aggregation functions (sum and mean). Another study that tests the effect of training with targeted versus untargeted PGD is presented in Appendix B. Altogether, we conclude that the majority of the gain in natural and robust accuracy is obtained by DBAT.

Numerical instability. The loss function calculates the log over the max on the Softmax probabilities. This additional log may cause numerical instability, as discussed in Appendix G.2 of [71]. To demonstrate that our method does not suffer from numerical instability, we conducted two additional experiments:

- We replaced the max with LogSumExp (LSE) which should be more stable. Changes in results were within a standard deviation of \( \pm 0.22 \) from the original reported results.
- We ran both AA and PGD-20 with 5 random restarts (within epsilon) and calculated mean and std. All results were within a standard deviation of \( \pm 0.2 \).

Model complexity and training time overhead. We acknowledge the fact that DBAT presents additional complexity to the model. However, keeping in mind that only the output of the last final fully connected layer is doubled, the additional model complexity is minor in most of the cases. Specifically, with WRN-32-10 on CIFAR-100, DBAT introduces 64k additional parameters, which sums up to an additional 0.13% of the total parameters. Additionally, we also analyzed the training time overhead of our approach compared to the well-known TRADES. Overall, DBAT has a minor overhead of up to 2-3% for models with class numbers ranging from 10 to 100.

5.8 Adaptive vs. Non-adaptive Attacks

In §5.1 we stated that the most powerful white-box adversary is one who has access to both model parameters and to the projection function at any given time during inference. This adversary, which we used throughout the paper and termed inference real-time access adversary, possesses all possible capabilities and may also be thought of as a “Perfect-Knowledge Adversary”. That is, the adversary has access to the model parameters, and more importantly, to the projection function the defender is using at each given time during inference — and thus can utilize the same projection function while attacking. Throughout the paper, we compare primarily against this unrealistically powerful adaptive adversary. Although this adversary is not the most realistic one, here we wish to demonstrate that this kind of adversary is indeed the most powerful adversary. To do so, we compare the results with two different white-box adversaries. The first adversary will be referred to as the model parameters adversary. It is assumed to possess access to the model parameters, but not to the inference projection function — meaning that the adversary can optimize the network parameters, but is not utilizing the inference projection function (e.g., max) in the attack optimization process. The model parameters adversary illustrates how the attack success rate is influenced by the attacker’s adaptive knowledge (or lack thereof) about the defender’s projection function. The second, most realistic adversary, termed model parameters + projection function access adversary, knows that the defender is utilizing a projection function, but does
Table 6: Natural and Auto-Attack (AA) $\ell_\infty$ robustness on CIFAR-10, CIFAR-100 and SVHN, against adversaries with different capabilities. Model parameters + projection function access refers to the adversary capabilities to access both model parameters and inference projection function. Inference real-time access refers to the adversary that can access not only to the entire network parameters but also to the defender’s system and projection function at any given time during inference. Model parameters refers to an adversary that gains access only to the model parameters, but not to the inference time projection function.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Adversary access capabilities</th>
<th>Natural Acc.</th>
<th>Robust Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10</td>
<td>Model parameters access</td>
<td>95.01</td>
<td>50.31</td>
</tr>
<tr>
<td></td>
<td>Model parameters + projection function access</td>
<td>47.82</td>
<td>40.08</td>
</tr>
<tr>
<td></td>
<td>Inference real-time access (model params. + inference-time access to projection func.)</td>
<td>40.08</td>
<td></td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>Model parameters access</td>
<td>75.18</td>
<td>23.16</td>
</tr>
<tr>
<td></td>
<td>Model parameters + projection function access</td>
<td>20.87</td>
<td>18.17</td>
</tr>
<tr>
<td></td>
<td>Inference real-time access (model params. + inference-time access to projection func.)</td>
<td>18.17</td>
<td></td>
</tr>
<tr>
<td>SVHN</td>
<td>Model parameters access</td>
<td>96.86</td>
<td>56.60</td>
</tr>
<tr>
<td></td>
<td>Model parameters + projection function access</td>
<td>48.58</td>
<td>40.49</td>
</tr>
<tr>
<td></td>
<td>Inference real-time access (model params. + inference-time access to projection func.)</td>
<td>40.49</td>
<td></td>
</tr>
</tbody>
</table>

Table 7: Natural and Auto-Attack (AA) results against the strongest, inference real-time access adversary, using two additional aggregation functions: sum and mean.

<table>
<thead>
<tr>
<th>Agg. function</th>
<th>Acc.</th>
<th>CIFAR-10</th>
<th>SVHN</th>
<th>CIFAR-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum</td>
<td>Natural</td>
<td>95.01</td>
<td>96.85</td>
<td>75.07</td>
</tr>
<tr>
<td></td>
<td>AA</td>
<td>39.81</td>
<td>36.00</td>
<td>18.08</td>
</tr>
<tr>
<td>Mean</td>
<td>Natural</td>
<td>95.01</td>
<td>96.85</td>
<td>75.07</td>
</tr>
<tr>
<td></td>
<td>AA</td>
<td>39.93</td>
<td>35.84</td>
<td>18.49</td>
</tr>
</tbody>
</table>

not have inference-time access to the defender’s choices at any given time during inference, and therefore has to conjecture the projection function from common projection functions (mean, softmax, maximum, sum, etc.) while attacking. Both adversaries — model parameters and model parameters + projection function access — are more realistic than the inference real-time access adversary, since the projection function is not part of the optimization. For example, the trained model can be published at model zoos, while the projection function does not. Alternatively, the defender can randomize or change the projection function at any time during inference.

Table 6 presents results under the three adversaries’ settings. As can be seen, since the model parameters adversary does not utilize the knowledge about the projection function, the attack against DBAT becomes much less effective, and as a consequence the attack success rate decreases, i.e., model robustness increases.

6 Conclusion

In this paper, we demonstrate the advantage of treating the clean and adversarially perturbed examples as belonging to separate classes, instead of insisting that the classifier “stretch” a single class to accommodate them both. With this new idea in mind, we proposed Double Boundary Adversarial Training (DBAT). Our extensive evaluation illustrates the ability of DBAT to achieve state-of-the-art results under various tasks such as black-box PGD attacks, natural corruptions robustness, and unforeseen adversaries (e.g., $\ell_2$-PGD, $\ell_1$-PGD, and DeepFool). That said, our aim is not to compete with the state-of-the-art in robustness across the board. Rather, we wish to equip models with a significant level of robustness, while only incurring a minor or negligible degradation to their original natural accuracy. Therefore, the main benefit of DBAT is its ability to reach optimal or near-optimal natural accuracy while achieving significant robustness, even against strong adversaries. This ability makes DBAT applicable for real-world applications (e.g., healthcare, autonomous vehicles, and security systems) that cannot sacrifice much of their natural accuracy.

7 Ethical Considerations

The existence of adversarial examples points to a basic weakness of deep neural networks. With the deployment of AI in safety-critical systems, such as security systems, medical diagnosis, and autonomous driving, it is at a premium to build systems that are robust, at least to some extent, against such attacks. However, the gain of robustness is usually at the expense of the systems’ natural accuracy. In order for real-world, safety-critical systems to adopt robust models, we need to make sure that the degradation in the natural accuracy is minor. For this reason, we suggested DBAT, which achieved a significant level of robustness without sacrificing much of the natural accuracy, and hope that it will help real-world
applications adopt robust models.

Having said that, DBAT still has its limitations: adversarial training is an expensive training method that requires extra computations when compared to vanilla training. Moreover, DBAT achieves significant level of robustness, but it does not eliminate it completely, and one should take it under consideration when deploying such methods.

References


A Full Experimental Setup Details

We conduct our experiments on CIFAR10, CIFAR100 [39] and SVHN [48]. We use the wide residual network (WRN-34-10) [92] architecture for CIFAR-10 and CIFAR-100, and the PreAct ResNet-18 [33] for SVHN. Following [58], we apply early-stopping based on a validation set. The batch size is set to 128, weight decay is set to $7e^{-4}$. We train the model for 200 epochs with an initial learning rate of 0.1. For CIFAR datasets, the learning rate is decayed by a factor of 10 at iterations 50 and 150. For SVHN, the learning rate is decayed by a factor of 10 at iterations 50 and 75. Natural images are padded with 4-pixel padding with 32-random crop and random horizontal flip. Furthermore, all methods are trained using SGD with momentum 0.9. We combine Stochastic Weight Averaging (SWA) [36], exponential moving average on the model weights during training steps, within our training process. SWA was shown to be effective in training robust models [14, 57], due to its temporal ensemble effect, and the ability to smooth the weights.

B Training with Targeted vs. Untargeted PGD

As previously stated, using targeted-PGD on random target labels helps to better generalize the adversarial classes. In Table 8, we show the comparison of the results between random targeted PGD (T-PGD), Least-Likely (least likely label based on the model decision) targeted PGD, and untargeted PGD. Using Least-Likely T-PGD achieves optimal natural accuracy, at the cost of a decrease in robust accuracy. Using untargeted PGD, we experience better AA robustness, at the cost of a decrease in natural accuracy.

Table 8: Natural accuracy and Auto-Attack robust accuracy against the strongest adversary, Inference real-time access, on CIFAR-10.

<table>
<thead>
<tr>
<th>ATTACK TYPE</th>
<th>NATURAL ACC.</th>
<th>AA</th>
</tr>
</thead>
<tbody>
<tr>
<td>RANDOM T-PGD</td>
<td>95.01</td>
<td>40.08</td>
</tr>
<tr>
<td>LEAST-LIKELY T-PGD</td>
<td><strong>95.67</strong></td>
<td>34.42</td>
</tr>
<tr>
<td>UNTARGETED PGD</td>
<td>92.18</td>
<td><strong>44.13</strong></td>
</tr>
</tbody>
</table>

Finally, we conducted a case study experiment on CIFAR-10 for two other training methods, AT [45] and TRADES [96], in order to test if random targeted PGD can help improve other methods’ results as much as it helped our method. For AT, Auto-Attack robust accuracy was 45.94%, a decrease of 5.58% in robustness, with a small improvement of almost 3% in natural accuracy. For TRADES, Auto-Attack robust accuracy was 52.32%, a decrease of 0.76% in robustness, with a small improvement of 1.07% in natural accuracy. We can conclude that random targeted PGD degrades robustness for the other tested methods with only marginal improvement in natural accuracy.

C Rademacher Analysis

Rademacher complexity. As mentioned in Section 3.4, the VC-dimension is impractical in analyzing deep neural networks with a large number of weights. We will now argue that the thrust of our point continues to hold for the Rademacher complexity as well, which is far more practical as far as providing finite-sample generalization bounds [8, 91]. We assume a basic familiarity with this notion and refer the reader to [46] for background. For a brief recap, if $F$ is a collection of functions mapping some set $\Omega$ to $\mathbb{R}$, and $X_1, \ldots , X_n$ is sampled iid from some distribution on $\Omega$, then the (empirical) Rademacher complexity is defined by

$$R_n(F; X_1, \ldots , X_n) = \mathbb{E} \sup_{\sigma \in \mathcal{F}} \frac{1}{n} \sum \sigma_i f(X_i),$$ (6)

where expectation is over the $\sigma_i$, which are iid Rademacher variables (i.e., $\mathbb{P}(\sigma_i = 1) = \mathbb{P}(\sigma_i = -1) = 1/2$). It is a classic fact [46, Theorem 3.5] that the Rademacher complexity upper-bounds the generalization error: if $X_1, \ldots , X_n$ is an iid sample,
then
\[\text{err}(h) \leq \text{err}(h) + R_n(H; X_1, \ldots, X_n) + 3\sqrt{\frac{\log(2/\delta)}{2n}} \] (7)
holds uniformly over all \( h \in H \) with probability\(^1\) at least \( 1 - \delta \).

It is always the case [22, Theorem 3.2, Theorem 4.3] that
\[R_n(H; X_1, \ldots, X_n) \leq c \sqrt{\frac{V}{n}}, \] (8)
where \( V \) is the VC-dimension of \( H \) and \( c \) is a universal constant. Observe that the left-hand side of (8) is sensitive to the sampling distribution of \( X_i \) (and can be arbitrarily small for very concentrated distributions), while the right-hand side is distribution-free — and hence the bound in (8) can be rather loose. In situations where the VC-dimension is very large, the more delicate Rademacher analysis gives much tighter bounds than VC theory [8, 91].

We will make use of a recent Rademacher \( \ell_\infty \) vector contraction result:

**Theorem 1.** [27] Let \( F \) be a \( \mathbb{R}^\ell \)-valued function class, such that the coordinate projection class is denoted by \( F_j = \{ w \mapsto f(w)_j \mid f \in F \} \), for \( 1 \leq j \leq \ell \). Let \( \{ \phi_k \}_{k \leq n} \) be a sequence of functions such that each \( \phi_k \) is \( L \)-Lipschitz with respect to \( \ell_\infty \) norm. For any \( \alpha > 0 \), there exists a constant \( C_\alpha > 0 \) such that if \( \| \phi_k(f(w)) \|_{\ell_\infty} \leq B \), then it holds for any sequence \( w = (w_1, \ldots, w_n) \),
\[
R_n(\phi \circ F | w) := E_{\sigma} \left[ \sup_{f \in F} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \phi_k(f_i(w_i)) \right] 
\leq C_\alpha L \sqrt{\ell} \cdot \max_{\ell \in [\ell]} \sup_{a = [a_1, \ldots, a_\ell]} R_n(F_i | a) \cdot \log^\frac{1}{2} + \alpha \left( \frac{Bn}{\max_{\ell \in [\ell]} \sup_{a = [a_1, \ldots, a_\ell]} R_n(F_i | a)} \right).
\]

We observe, as in [6, Theorem 5.1], that \( \max(x_1, \ldots, x_\ell) \) is a \( 1 \)-Lipschitz function with respect to the \( \ell_\infty \) norm. It follows (taking \( \alpha = 1/2 \) and specializing the argument of [6, Theorem 5.1] to our simpler case), that if \( H^{1,\ell} \) is the \( \ell \)-fold union of \( H \) — that is, every \( H' \in H^{1,\ell} \) can be expressed as the union of some \( \ell \) members of \( H \) — then we have:
\[
R_n(H^{1,\ell}; X_1, \ldots, X_n) \leq C_\alpha \sqrt{\ell} \max_{\ell \in [\ell]} R_n(H) \log^\frac{1}{2} \frac{n}{R_n(H)}, \] (9)
where \( C \) is a universal constant and \( \tilde{R}_n(H) := \sup_{(x_1, \ldots, x_\ell) \in \Omega} R_n(H; x_1, \ldots, x_\ell) \). Since the Rademacher complexity of the \( \ell \)-fold union grows roughly as \( \sqrt{\ell} \), it may often be advantageous to re-analyze complex hypotheses as unions of simple ones — just as we concluded for the VC-dimension in §4. Thus, we can apply (7) to the two competing approaches: (i) when learning \( \ell \) “simple” classifiers from the class \( H \), and (ii) when learning \( k \) “complex” classifiers from the class \( H^{1,\ell} \). Ignoring logarithmic factors, and treating \( \delta \) as fixed, we can compare the bound of order roughly \( R_n(H) + \sqrt{\log(\ell k)/n} \) when splitting the classifiers (the \( \ell k \) inside the \( \log \) is from the union bound) and roughly \( \sqrt{R_n(H) + \log(k)/n} \) without splitting (there are only \( k \) classes but each incurred a \( \sqrt{\ell} \) factor from (9)). For simple hypothesis classes \( H \) (i.e., those with a low Rademacher complexity), this again demonstrates the advantage of splitting.

**D Distance to Decision Boundary**

In the following experiment, we wish to demonstrate that the conceptual illustration that we’ve drawn can exist in real datasets. To do so, we need to demonstrate that most clean examples are relatively close to the decision boundary, in a distance that is half of the perturbation size \( (\epsilon/2 = 4/255) \), so that it will support the way we’ve drawn the intuition in 1, where examples “switch” places after the attack. In the following experiment, we used SVHN with a naturally trained model using PreAct ResNet-18, and 1000 random examples from SVHN’s test set. To calculate the distance, we followed the distance estimation suggested in [34]. We estimate the distance to a decision boundary in a sample of random directions in the model’s input space, starting from a given input point. In each direction, we estimate the distance to a decision boundary by computing the model’s prediction on perturbed inputs at points along the direction and increase the random directions by a magnitude factor (0.002) if the prediction does not change in any of the directions. We perform this search over a set of 1,000 random orthogonal directions. Results are present in Figure 10.

**Figure 10:** Histogram of the estimated distance from the decision boundary.