# REVISITING DEEPFOOL: GENERALIZATION AND IMPROVEMENT

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Deep neural networks have been known to be vulnerable to adversarial examples, which are inputs that are modified slightly to fool the network into making incorrect predictions. This has led to a significant amount of research on evaluating the robustness of these networks against such perturbations. One particularly important robustness metric is the robustness to minimal $\ell_2$ adversarial perturbations. However, existing methods for evaluating this robustness metric are either computationally expensive or not very accurate. In this paper, we introduce a new family of adversarial attacks that strike a balance between effectiveness and computational efficiency. Our proposed attacks are generalizations of the well-known DeepFool (DF) attack, while they remain simple to understand and implement. We demonstrate that our attacks outperform existing methods in terms of both effectiveness and computational efficiency. Our proposed attacks are also suitable for evaluating the robustness of large models and can be used to perform adversarial training (AT) to achieve state-of-the-art robustness to minimal $\ell_2$ adversarial perturbations.

## 1 INTRODUCTION

Deep learning has achieved breakthrough improvement in numerous tasks and has developed as a powerful tool in various applications, including computer vision Long et al. (2015) and speech processing Mikolov et al. (2011). Despite their success, deep neural networks are known to be vulnerable to adversarial examples, carefully perturbed examples perceptually indistinguishable from original samples Szegedy et al. (2013). This can lead to a significant disruption of the inference result of deep neural networks. It has important implications for safety and security-critical applications of machine learning models.

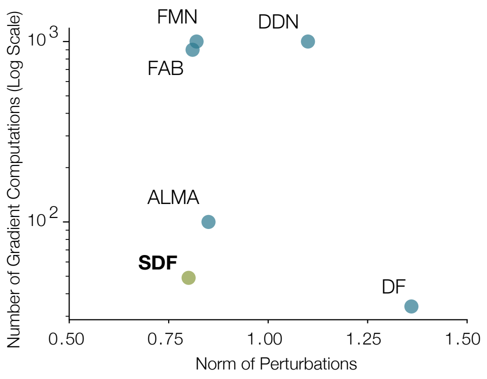


Figure 1: The average number of gradient computations vs the mean $\ell_2$-norm of perturbations. It shows that our novel fast and accurate method, SDF, outperforms other minimum-norm attacks. SDF finds significantly smaller perturbations compared to DF, with only a small increase in computational cost. SDF also outperforms other algorithms in optimality and speed. The numbers are taken from Table 5.

Our goal in this paper is to introduce a parameter-free and simple method for accurately and reliably evaluating the adversarial robustness of deep networks in a fast and geometrically-based fashion. Most of the current attack methods rely on general-purpose optimization techniques, such as Projected Gradient Descent (PGD) Madry et al. (2017) and Augmented Lagrangian Rony et al. (2021), which are oblivious to the geometric properties of models. However, deep neural networks' robustness to adversarial perturbations is closely tied to their geometric landscape Dauphin et al. (2014); Poole et al. (2016). Given this, it would be beneficial to exploit such properties when designing and implementing adversarial attacks. This allows to create more effective and computationally efficient attacks on classifiers. Formally, for a given classifier $\hat{k}$ and input $\boldsymbol{x}$, we define an adversarial perturbation as the minimal perturbation $\boldsymbol{r}$ that is sufficient to change the estimated label $\hat{k}(\boldsymbol{x})$:

$$\Delta(\boldsymbol{x}; \hat{k}) := \min_{\boldsymbol{r}} \|\boldsymbol{r}\|_2 \text{ s.t } \hat{k}(\boldsymbol{x}+\boldsymbol{r}) \neq \hat{k}(\boldsymbol{x}). \tag{1}$$

DeepFool (DF) Moosavi-Dezfooli et al. (2016) was among the earliest attempts to exploit the "excessive linearity" Goodfellow et al. (2014) of deep networks to find minimum-norm adversarial perturbations. However, more sophisticated attacks were later developed that could find smaller perturbations at the expense of significantly greater computation time.

In this paper, we exploit the geometric characteristics of minimum-norm adversarial perturbations to design a family of fast yet simple algorithms that achieves a better trade-off between computational cost and accuracy in finding $\ell_2$ adversarial perturbations (see Fig. 1). Our proposed algorithm, guided by the characteristics of the optimal solution to Eq. (1), enhances DF to obtain smaller perturbations, while maintaining simplicity and computational efficiency that are only slightly inferior to those of DF. Our main contributions are summarized as follows:

- We introduce a novel family of fast yet accurate algorithms to find minimal adversarial perturbations. We extensively evaluate and compare our algorithms with state-of-the-art attacks in various settings.
- Our algorithms are developed in a systematic and well-grounded manner, based on theoretical analysis.
- We further improve the robustness of state-of-the-art image classifiers to minimum-norm adversarial attacks via adversarial training on the examples obtained by our algorithms.
- We significantly improve the time efficiency of the state-of-the-art Auto-Attack (AA) Croce & Hein (2020b) by adding our proposed method to the set of attacks in AA.

**Related works.** It has been observed that deep neural networks are vulnerable to adversarial examples Szegedy et al. (2013); Moosavi-Dezfooli et al. (2016); Goodfellow et al. (2014). To exploit this vulnerability, a range of methods have been developed for generating adversarial perturbations for image classifiers. These attacks occur in two settings: white-box, where the attacker has complete knowledge of the model, including its architecture, parameters, defense mechanisms, etc.; and black-box, where the attacker's knowledge is limited, mostly relying on input queries to observe outputs Chen et al. (2020); Rahmati et al. (2020). Further, adversarial attacks can be broadly categorized into two categories: bounded-norm attacks (such as FGSM Goodfellow et al. (2014) and PGD Madry et al. (2017)) and minimum-norm attacks (such as DF and C&W Carlini & Wagner (2017)) with the latter aimed at solving Eq. (1). In this work, we specifically focus on white-box minimum $\ell_2$-norm attacks.

The authors in Szegedy et al. (2013) studied adversarial examples by solving a penalized optimization problem. The optimization approach used in Szegedy et al. (2013) is complex and computationally inefficient; therefore, it cannot scale to large datasets. The method proposed in Goodfellow et al. (2014) applied a single-step of the input gradient to generate adversarial examples efficiently. DF was the first method to seek minimum-norm adversarial perturbations, employing an iterative approach. It linearizes the classifier at each step to estimate the minimal adversarial perturbations efficiently. C&W attack Carlini & Wagner (2017) transform the optimization problem in Szegedy et al. (2013) into an unconstrained optimization problem. C&W leverages the first-order gradient-based optimizers to minimize a balanced loss between the norm of the perturbation and misclassification confidence. Inspired by the geometric idea of DF, FAB Croce & Hein (2020a) presents an approach to minimize the norm of adversarial perturbations by employing complex projections and approximations while maintaining proximity to the decision boundary. By utilizing gradients to estimate the local geometry of the boundary, this method formulates minimum-norm optimization without the need for tuning a weighting term. DDN Rony et al. (2019) uses projections on the $\ell_2$-ball for a given perturbation budget $\epsilon$. FMN Pintor et al. (2021) extends the DDN attack to other $\ell_p$-norms. By formulating (1) with Lagrange's method, ALMA Rony et al. (2021) introduced a framework for finding adversarial examples for several distances.

## 2 DeepFool (DF) and Minimal Adversarial Perturbations

In this section, we first discuss the geometric interpretation of the minimum-norm adversarial perturbations, i.e., solutions to the optimization problem in Eq. (1). We then examine DF to demonstrate why it may fail to find the optimal minimum-norm perturbation. Then in the next section, we introduce our proposed method that exploits DF to find smaller perturbations.

Let $f : \mathbb{R}^d \to \mathbb{R}^C$ denote a $C$-class classifier, where $f_k$ represents the classifier's output associated to the $k$th class. Specifically, for a given datapoint $\boldsymbol{x} \in \mathbb{R}^d$, the estimated label is obtained by $\hat{k}(\boldsymbol{x}) = \text{argmax}_k f_k(\boldsymbol{x})$, where $f_k(\boldsymbol{x})$ is the $k^{\text{th}}$ component of $f(\boldsymbol{x})$ that corresponds to the $k^{\text{th}}$ class. Note that the classifier $f$ can be seen as a mapping that partitions the input space $\mathbb{R}^d$ into classification regions, each of which has a constant estimated label (i.e., $\hat{k}(.)$ is constant for each such region). The decision boundary $\mathscr{B}$ is defined as the set of points in $\mathbb{R}^d$ such that $f_i(\boldsymbol{x}) = f_j(\boldsymbol{x}) = \max_k f_k(\boldsymbol{x})$ for some distinct $i$ and $j$.

Additive $\ell_2$-norm adversarial perturbations are inherently related to the geometry of the decision boundary. More formally, Let $\boldsymbol{x} \in \mathbb{R}^d$, and $\boldsymbol{r}^*(\boldsymbol{x})$ be the minimal adversarial perturbation defined as the minimizer of Eq. (1). Then $\boldsymbol{r}^*(\boldsymbol{x})$, 1) is orthogonal to the decision boundary of the classifier $\mathscr{B}$, and 2) its norm $\|\boldsymbol{r}^*(\boldsymbol{x})\|_2$ measures the Euclidean distance between $\boldsymbol{x}$ and $\mathscr{B}$, that is $\boldsymbol{x} + \boldsymbol{r}^*$ lies on $\mathscr{B}$. We aim to investigate whether the perturbations generated by DF satisfy the aforementioned two conditions. Let $\boldsymbol{r}_{\text{DF}}$ denote the perturbation found by DF for a datapoint $\boldsymbol{x}$. We expect $\boldsymbol{x} + \boldsymbol{r}_{\text{DF}}$ to lie on the decision boundary. Hence, if $\boldsymbol{r}$ is the minimal perturbation, for all $0 < \gamma < 1$, we expect the perturbation $\gamma \boldsymbol{r}$ to remain in the same decision region as of $\boldsymbol{x}$ and thus fail to fool the model.

In Fig. 2 (top-left), we consider the fooling rate of $\gamma\, \boldsymbol{r}_{\text{DF}}$ for $0.2 < \gamma < 1$. For a minimum-norm perturbation, we expect an immediate sharp decline for $\gamma$ close to one. However, in Fig. 2 (top-left) we cannot observe such a decline (a sharp decline happens close to $\gamma = 0.9$, not 1). This is a confirmation that DF typically finds an overly perturbed point. One potential reason for this is the fact that DF stops when a misclassified point is found, and this point might be an overly perturbed one within the adversarial region, and not necessarily on the decision boundary.

Now, let us consider the other characteristic of the minimal adversarial perturbation. That is, the perturbation should be orthogonal to the decision boundary. We measure the angle between the found perturbation $\boldsymbol{r}_{\text{DF}}$ and the normal vector orthogonal to the decision boundary ($\nabla f(\boldsymbol{x} + \boldsymbol{r}_{\text{DF}})$). To do so, we first scale $\boldsymbol{r}_{\text{DF}}$ such that $\boldsymbol{x} + \gamma \boldsymbol{r}_{\text{DF}}$ lies on the decision boundary. It can be simply done via performing a line search along $\boldsymbol{r}_{\text{DF}}$. We then compute the cosine of the angle between $\boldsymbol{r}_{\text{DF}}$ and the normal to the decision boundary at $\boldsymbol{x} + \gamma \boldsymbol{r}_{\text{DF}}$ (this angle is denoted by $\cos(\alpha)$). A necessary condition for $\gamma \boldsymbol{r}_{\text{DF}}$ to be an optimal perturbation is that it must be parallel to the normal vector of the decision boundary (Figure 6 in the Appendix). In Fig. 2 (top-right) , we show the distribution of cosine of this angle. Ideally, we wanted this distribution to be accumulated around one. However, it clearly shows that this is not the case, which is a confirmation that $\boldsymbol{r}_{\text{DF}}$ is not necessarily the minimal perturbation.

## 3 SUPERDEEPFOOL: EFFICIENT ALGORITHMS TO FIND MINIMAL PERTURBATIONS

In this section, we propose a new class of methods that modifies DF to address the aforementioned challenges in the previous section. The goal is to maintain the desired characteristics of DF, i.e., computational efficiency and the fact that it is parameter-free while finding smaller adversarial perturbations. We achieve this by introducing an additional projection step which its goal is to steer the direction of perturbation towards the optimal solution of Eq. (1).

Let us first briefly recall how DF finds an adversarial perturbations for a classifier $f$. Given the current point $\boldsymbol{x}_i$, DF updates it according to the following equation:

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \frac{f(\boldsymbol{x}_i)}{\|\nabla f(\boldsymbol{x}_i)\|_2^2} \nabla f(\boldsymbol{x}_i). \tag{2}$$

Here the gradient is taken w.r.t. the input. The intuition is that, in each iteration, DF finds the minimum perturbation for a linear classifier that approximates the model around $\boldsymbol{x}_i$. The below proposition shows that under certain conditions, repeating this update step eventually converges to a point on the decision boundary.

**Proposition 1** *Let the binary classifier $f : \mathbb{R}^d \to \mathbb{R}$ be continuously differentiable and its gradient $\nabla f$ and $f$ are $L^{'}$-Lipschitz. For a given input sample $\boldsymbol{x}_0$, suppose $B(\boldsymbol{x}_0, \epsilon)$ is a ball centered around*

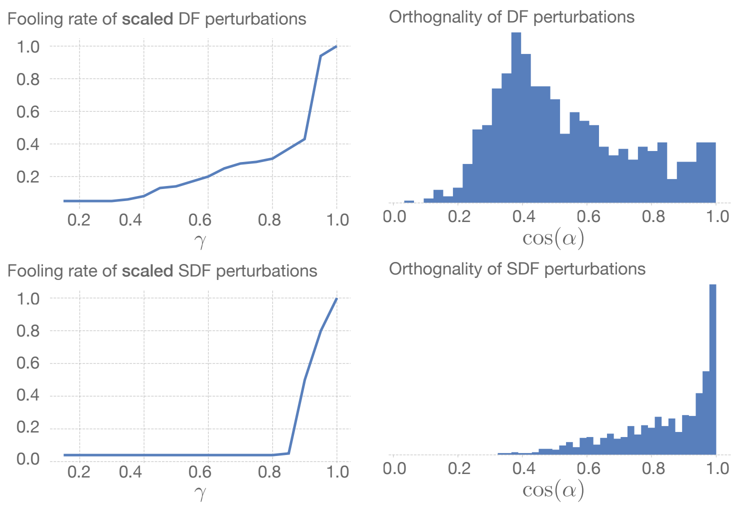


Figure 2: **(Left)** we generated 1000 images with one hundred $\gamma$ between zero and one, and the fooling rate of the DeepFool (top) and SuperDeepFool (bottom) is reported. This experiment is done on the CIFAR10 dataset and ResNet18 model. **(Right)** histogram of the cosine angle between the normal to the decision boundary and the perturbation vector obtained by DeepFool (top) and SuperDeepFool (bottom) has been showed.

*$\boldsymbol{x}_0$ with radius $\epsilon$, such that there exists $\boldsymbol{x} \in B(\boldsymbol{x}_0, \epsilon)$ that $f(\boldsymbol{x}) = 0$. If $\|\nabla f\|_2 \geq \zeta$ for all $\boldsymbol{x} \in B$ and $\epsilon < \frac{\zeta^2}{L'^2}$, then DF iterations converge to a point on the decision boundary.*

*Proof: We defer the proof to the Appendix.*

Notice while the proposition guarantees the perturbed sample to lie on the decision boundary, it does not state anything about the orthogonality of the perturbation to the decision boundary.

To find perturbations that are more aligned with the normal to the decision boundary, we introduce an additional projection step that steers the perturbation direction towards the optimal solution of Eq. (1). Formally, the optimal perturbation, $\boldsymbol{r}^*$, and the normal to the decision boundary at $\boldsymbol{x}_0 + \boldsymbol{r}^*$, $\nabla f(\boldsymbol{x}_0 + \boldsymbol{r}^*)$, should be parallel. Equivalently, $\boldsymbol{r}^*$ should be a solution of the following maximization problem:

$$\max_{\boldsymbol{r}} \frac{\boldsymbol{r}^\top \nabla f(\boldsymbol{x}_0 + \boldsymbol{r})}{\|\nabla f(\boldsymbol{x}_0 + \boldsymbol{r})\| \|\boldsymbol{r}\|}, \tag{3}$$

which is the cosine of the angle between $\boldsymbol{r}$ and $\nabla f(\boldsymbol{x}_0 + \boldsymbol{r})$. A necessary condition for $\boldsymbol{r}^*$ to be a solution of Eq. (3) is that the projection of $\boldsymbol{r}^*$ on the subspace orthogonal to $\nabla f(\boldsymbol{x}_0 + \boldsymbol{r}^*)$ should be zero. Then, $\boldsymbol{r}^*$ can be seen as a fixed point of the following iterative map:

$$\boldsymbol{r}_{i+1} = T(\boldsymbol{r}_i) = \frac{\boldsymbol{r}_i{}^\top \nabla f(\boldsymbol{x}_0 + \boldsymbol{r}_i)}{\|\nabla f(\boldsymbol{x}_0 + \boldsymbol{r}_i)\|} \cdot \frac{\nabla f(\boldsymbol{x}_0 + \boldsymbol{r}_i)}{\|\nabla f(\boldsymbol{x}_0 + \boldsymbol{r}_i)\|}. \tag{4}$$

The scalar multiplier on the right-hand side of Eq. (4) represents the norm of the projection of the vector $\boldsymbol{r}_i$ along the gradient direction. The following proposition shows that this iterative process can converge to a solution of Eq. (3).

**Proposition 2** *For a differentiable $f$ and a given $\mathbf{r}_0$, $\boldsymbol{r}_i$ in the iterations Eq. (4) either converge to a solution of Eq. (3) or a trivial solution (i.e., $\boldsymbol{r}_i \to 0$).*

*Proof: We defer the proof to the Appendix.*

### 3.1 A Family of Adversarial Attacks

**Algorithm 1:** SDF $(m,n)$ for binary classifiers

```
Input: image $x_0$, classifier $f$, $m$, and $n$.
Output: perturbation $r$
1  Initialize: $x \leftarrow x_0$
2  while $\mathrm{sign}(f(x)) = \mathrm{sign}(f(x_0))$ do
3      repeat $m$ times
4          $x \leftarrow x - \frac{|f(x)|}{\|\nabla f(x)\|_2^2}\nabla f(x)$
5      end
6      repeat $n$ times
7          $x \leftarrow x_0 + \frac{(x-x_0)^\top \nabla f(x)}{\|\nabla f(x)\|^2}\nabla f(x)$
8      end
9  end
10 return $r = x - x_0$
```

Finding minimum-norm adversarial perturbations can be seen as a multi-objective optimization problem, where we want $f(\boldsymbol{x}+\boldsymbol{r}) = 0$ and the perturbation $\boldsymbol{r}$ to be orthogonal to the decision boundary. So far we have seen that DF finds a solution satisfying the former objective and the iterative map Eq. (4) can be used to find a solution for the latter. A natural approach to satisfy both objectives is to ***alternate*** between these two iterative steps, namely Eq. (2) and Eq. (4). We propose a family of adversarial attack algorithms, coined SuperDeepFool, by varying how frequently we alternate between these two steps. We denote this family of algorithms with SDF$(m, n)$, where $m$ is the number of DF steps Eq. (2) followed by $n$ repetition of the projection step Eq. (4). This process is summarized in Algorithm 1. One interesting case is SDF$(\infty, 1)$ which, in each iteration, continues DF steps till a point on the decision boundary is found and then applies the projection step. This particular case has a resemblance with the strategy used in Rahmati et al. (2020) to find black-box adversarial perturbations. This algorithm can be interpreted as iteratively approximating the decision boundary with a hyperplane and then analytically calculating the minimal adversarial perturbation for a linear classifier for which this hyperplane is the decision boundary. It is justified by the observation that the decision boundary of state-of-the-art deep networks has a small mean curvature around data samples Fawzi et al. (2017; 2018). A geometric illustration of this procedure is shown in Figure 3.

### 3.2 SDF Attack

We empirically compare the performance of SDF$(m, n)$ for different values of $m$ and $n$ in Section 4.1. Interestingly, we observe that we get better attack performance when we apply several DF steps followed by a single projection. Since the standard DF typically finds an adversarial example in less than four iterations for state-of-the-art image classifiers, one possibility is to continue DF steps till an adversarial example is found and then apply a single projection step. We simply call this particular version SDF$(\infty, 1)$ of our algorithm SDF, which we will extensively evaluate in Section 7. SDF can be understood as a generic algorithm that can also work for the multi-class case by simply substituting the first inner loop of Algorithm 1 with the standard multi-class DF algorithm. The label of the obtained adversarial example determines the boundary on which the projection step will be performed. A summary of multi-class SDF is presented in Algorithm 2. Compared to the standard DF, this algorithm has an additional projection step. We will see later that such a simple modification leads to significantly smaller perturbations.

**Algorithm 2:** SDF for multi-class classifiers

```
Input: image $x_0$, classifier $f$.
Output: perturbation $r$
1  Initialize: $x \leftarrow x_0$
2  while $\hat{k}(x) = \hat{k}(x_0)$ do
3      $\widetilde{x} \leftarrow \mathtt{DeepFool}(x)$
4      $w \leftarrow \nabla f_{\hat{k}(\widetilde{x})}(\widetilde{x}) - \nabla f_{\hat{k}(x_0)}(\widetilde{x})$
5      $x \leftarrow x_0 + \frac{(\widetilde{x}-x_0)^\top w}{\|w\|^2} w$
6  end
7  return $r = x - x_0$
```

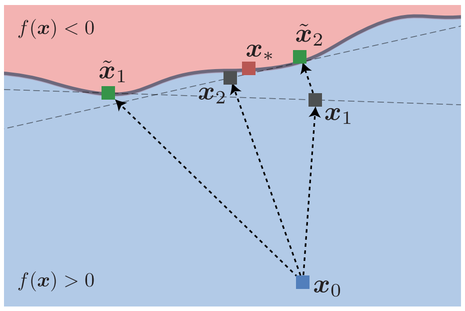


Figure 3: Illustration of two iterations of the SDF($\infty$,1) algorithm. Here $\mathbf{x}_0$ is the original data point and $\mathbf{x}_*$ is the minimum-norm adversarial example.

Table 1: The cosine of the angle between the perturbation vector ($\boldsymbol{r}$) and $\nabla f(\boldsymbol{x}+\boldsymbol{r})$. We performed this experiment on three models trained on CIFAR10 dataset.

| Attack | Models | | |
|---|---|---|---|
| | LeNet | ResNet18 | WRN-28-10 |
| DF | 0.89 | 0.14 | 0.21 |
| SDF (1,1) | 0.90 | 0.63 | 0.64 |
| SDF (1,3) | 0.88 | 0.61 | 0.62 |
| SDF (3,1) | **0.92** | 0.70 | 0.72 |
| SDF $(\infty, 1)$ | **0.92** | **0.72** | **0.80** |

Table 2: Comparison of the $\ell_2$-norm of perturbations for DF and SDF family algorithms. We performed this experiment on CIFAR10. We use the same model architecture and hyperparameters for training as in C&W and DDN.

| Attack | Mean-$\ell_2$ | Median-$\ell_2$ | Grads |
|---|---|---|---|
| DF | 0.17 | 0.15 | **14** |
| SDF (1,1) | 0.14 | 0.13 | 22 |
| SDF (1,3) | 0.16 | 0.14 | 26 |
| SDF (3,1) | 0.12 | 0.11 | 30 |
| SDF$(\infty, 1)$ | **0.11** | **0.10** | 32 |

## 4 Experimental Results

In this section, we conduct extensive experiments to demonstrate the effectiveness of our method in different setups and for several natural and adversarially trained networks. We first introduce our experimental settings, including datasets, models, and attacks. Next, we compare our method with state-of-the-art $\ell_2$-norm adversarial attacks in various settings, demonstrating the superiority of our simple yet fast algorithm for finding accurate adversarial examples. Moreover, we add SDF to the collection of attacks used in AutoAttack, and call the new set of attacks Auto-Attack++. This setup meaningfully speeds up the process of finding norm-bounded adversarial perturbations. We also demonstrate that a model adversarially training using the SDF perturbations becomes more robust compared to the models[1] trained using other minimum-norm attacks.

**Setup.** We test our algorithms on architectures trained on MNIST, CIFAR10, and ImageNet datasets. For MNIST, we use a robust model called IBP from Zhang et al. (2019) and naturally trained model called SmallCNN. For CIFAR10, we use three models: an adversarially trained PreActResNet-18 He et al. (2016b) from Rade & Moosavi-Dezfooli (2021), a regularly trained Wide ResNet 28-10 (WRN-28-10) from Zagoruyko & Komodakis (2016) and LeNet LeCun et al. (1999). These models are obtainable via the RobustBench library Croce et al. (2020). On ImageNet, we test the attacks on two ResNet-50 (RN-50) models: one regularly trained and one $\ell_2$ adversarially trained, obtainable through the robustness library Engstrom et al. (2019).

### 4.1 Comparison with DeepFool (DF)

In this part, we compare our algorithm in terms of orthogonality and size of the $\ell_2$-norm perturbations especially with DF. Assume $\boldsymbol{r}$ is the perturbation vector obtained by an adversarial attack. First, we measure the orthogonality of perturbations by measuring the inner product between $\nabla f(\boldsymbol{x}+\boldsymbol{r})$ and $\boldsymbol{r}$. As we explained in Section 2, a larger inner product between $\boldsymbol{r}$ and the gradient vector at $f(\boldsymbol{x}+\boldsymbol{r})$ indicates that the perturbation vector is closer to the optimal perturbation vector $\boldsymbol{r}^*$. We compare the orthogonality of different members of the SDF family and DF. The results are shown in Table 1. We observe that DF finds perturbations orthogonal to the decision boundary for low-complexity models such as LeNet, but fails to perform effectively when evaluated against more complex ones. In contrast, attacks from the SDF family consistently found perturbations with a larger cosine of the angle for all three models.

**Verifying optimality conditions for SDF.** We validate the optimality conditions of the perturbations generated by SDF using the procedure outlined in Section 2. Comparing Fig. 2 top (DF) and bottom (SDF) rows, it becomes evident that our approach effectively mitigates the two issues we previously highlighted for DF. Namely, the alignment of the perturbation with the normal to the decision boundary and the problem of over-perturbation. We can see that unlike DF, the cosine of the angle for SDF is more concentrated around one, which indicates that the SDF perturbations are more aligned with the normal to the decision boundary. Moreover, Fig. 2 (bottom-right) shows a sharper decline in the fooling rate (going down quickly to zero) when $\gamma$ decreases. This is consistent with

[1]We only compare to publicly available models.

Table 3: Performance for attacks on the MNIST dataset with IBP models. The numbers between parentheses indicate the number of iterations.

| Attack | FR | Median-$\ell_2$ | Grads |
|---|---|---|---|
| ALMA (1000) | **100** | **1.26** | 1 000 |
| ALMA (100) | 98.90 | 4.96 | 100 |
| DDN (1000) | 99.27 | 1.46 | 1 000 |
| DDN (100) | 94.34 | 1.97 | 100 |
| FAB (1000) | 99.98 | 3.34 | 10 000 |
| FAB (100) | 99.98 | 5.19 | 1 000 |
| FMN (1000) | 89.08 | 1.34 | 1 000 |
| FMN (100) | 67.80 | 2.14 | 100 |
| C&W | 4.63 | – | 90 000 |
| SDF | **100** | 1.37 | **52** |

Table 4: Performance of attacks on the CIFAR-10 dataset with WRN-28-10. The results on adversarially trained networks are deferred to Tables 20 and 11 of the Appendix.

| Attacks | FR | Median-$\ell_2$ | Grads |
|---|---|---|---|
| DF | 100 | 0.26 | **14** |
| ALMA | 100 | 0.10 | 100 |
| DDN | 100 | 0.13 | 100 |
| FAB | 100 | 0.11 | 100 |
| FMN | 97.3 | 0.11 | 100 |
| C&W | 100 | 0.12 | 90 000 |
| SDF | 100 | **0.09** | 25 |

our expectation for an accurate minimal perturbation attack. Table 2 demonstrates that SDF family outperforms DF in finding more accurate perturbations, particularly SDF($\infty$,1) which significantly outperforms DF at a small cost.

### 4.2 Comparison with minimum-norm attacks

We now compare SDF with state-of-the-art minimum $\ell_2$-norm attacks: C&W, FMN, DDN, ALMA, and FAB. For C&W, we use the same hyperparameters as in Rony et al. (2019). We use FMN, FAB, DDN, and ALMA with budgets of 100 and 1000 iterations and report the best performance. For a fair comparison, we clip the pixel-values of SDF-generated adversarial images to $[0, 1]$, consistent with the other minimum-norm attacks. We report the average number of gradient computations per sample, as these operations are computationally intensive and provide a consistent metric unaffected by hardware differences. We also provide a runtime comparison in Table 20 of the Appendix.

We evaluate the robustness of the IBP model, which is adversarially trained on the MNIST dataset, against state-of-the-art attacks in Table 3. We choose this robust model as it allows us to have a more nuanced comparison between different adversarial attacks. SDF and ALMA are the only attacks that achieve a 100% percent fooling rate against this model, whereas C&W is unsuccessful on most of the data samples. The fooling rates of the remaining attacks also degrade when evaluated with 100 iterations. For instance, FMN's fooling rate decreases from 89% to 67.8% when the number of iterations is reduced from 1000 to 100. This observation shows that, unlike SDF, selecting the ***necessary number of iterations*** is critical for the success of ***fixed-iteration*** attacks. Even for ALMA which can achieve a nearly perfect FR, decreasing the number of iterations from 1000 to 100 causes the median norm of perturbations to increase fourfold. In contrast, SDF is able to compute adversarial perturbations using the fewest number of gradient computations while still outperforming the other algorithms, except ALMA, in terms of the perturbation norm. However, it is worth noting that ALMA requires twenty times more gradient computations compared to SDF to achieve a marginal improvement in the perturbation norm.

Table 4 compares SDF with state-of-the-art attacks on the CIFAR10 dataset. The results show that state-of-the-art attacks have a similar norm of perturbations, but an essential point is the speed of attacks. SDF finds more accurate adversarial perturbation very quickly rather than other algorithms. We also evaluated all attacks on an adversarially trained model for the CIFAR10 dataset. SDF achieves smaller perturbations with half the gradient calculations than other attacks. SDF finds smaller adversarial perturbations for adversarially trained networks at a significantly lower cost than other attacks, requiring only 20% of FAB's cost and 50% of DDN's and ALMA's (see Tables 11 and 20 in the Appendix).

Table 5 demonstrates the performance of SDF on a naturally and adversarially trained models on ImageNet dataset. Unlike models trained on CIFAR10, where the attacks typically result in perturbations with similar norm, the differences between attacks are more nuanced for ImageNet

Table 5: Performance comparison of SDF with other state-of-the-art attacks for median $\ell_2$ on ImageNet dataset. FR columns show the fooling rates of attacks.

| | RN-50 | | | RN-50 (AT) | | |
|---|---|---|---|---|---|---|
| Attack | FR | Median-$\ell_2$ | Grads | FR | Median-$\ell_2$ | Grads |
| DF | 99.1 | 0.31 | **23** | 98.8 | 1.36 | **34** |
| ALMA | **100** | 0.10 | 100 | **100** | 0.85 | 100 |
| DDN | 99.9 | 0.17 | 1,000 | 99.7 | 1.10 | 1,000 |
| FAB | 99.3 | 0.10 | 900 | **100** | 0.81 | 900 |
| FMN | 99.3 | 0.10 | 1,000 | 99.9 | 0.82 | 1,000 |
| C&W | **100** | 0.21 | 82,667 | 99.9 | 1.17 | 52,000 |
| SDF | **100** | **0.09** | 37 | **100** | **0.80** | 49 |

Table 6: The comparison between $\ell_2$ robustness of our adversarial trained model and Rony et al. (2019) model. We perform this experiment on CIFAR10 dataset.

| Attack | SDF (Ours) | | DDN | |
|---|---|---|---|---|
| | Mean | Median | Mean | Median |
| DDN | 1.09 | 1.02 | 0.86 | 0.73 |
| FAB | 1.12 | 1.03 | 0.92 | 0.75 |
| FMN | 1.48 | 1.43 | 1.47 | 1.43 |
| ALMA | 1.17 | 1.06 | 0.84 | **0.71** |
| SDF | **1.06** | **1.01** | **0.81** | 0.73 |

Table 7: Average input curvature of WRN-28-10 models trained on CIFAR10 dataset, according to the measures proposed in Srinivas et al.. The second column shows the average spectral-norm of the Hessian w.r.t. input, $\|\nabla^2 f(\mathbf{x})\|_2$, and the third column shows the average of the same quantity normalized by the norm of the input gradient, $\mathcal{C}_f(\mathbf{x}) = \|\nabla^2 f(\mathbf{x})\|_2 / \|\nabla f(\mathbf{x})\|_2$. The standard deviation is denoted by numbers enclosed in brackets.

| Model | $\mathbb{E}_{\mathbf{x}}\|\nabla^2 f(\mathbf{x})\|_2$ | $\mathbb{E}_{\mathbf{x}}\mathcal{C}_f(\mathbf{x})$ |
|---|---|---|
| Standard | 600.06 (29.76) | 73.99 (6.62) |
| DDN AT | 2.86 (1.22) | 4.32 (2.91) |
| SDF AT (Ours) | **0.73** (0.08) | **1.66** (0.86) |

models. In particular, FAB, DDN, and FMN's performance degrades when the dataset changes. In contrast, SDF achieves smaller perturbations at a significantly lower cost than ALMA. This shows that the geometric interpretation of optimal adversarial perturbation, rather than viewing (1) as a non-convex optimization problem, can lead to an efficient solution. On the complexity aspect, the proposed approach is substantially faster than the other methods. In contrast, these approaches involve a costly minimization of a series of objective functions. We empirically observed that SDF converges in less than 5 or 6 iterations to a fooling perturbation; our observations show that SDF consistently achieves state-of-the-art minimum-norm perturbations across different datasets, models, and training strategies, while requiring the least number of gradient computations. This makes it readily suitable to be used as a baseline method to estimate the robustness of very deep neural networks on large datasets.

### 4.3 SDF Adversarial Training (AT)

In this section, we evaluate the performance of a model adversarially trained using SDF against minimum-norm attacks and AutoAttack. Our experiments provide valuable insights into the effectiveness of adversarial training with SDF and sheds light on its potential applications in building more robust models. Adversarial training requires computationally efficient attacks, making costly options such as C&W unsuitable. Therefore, an attack that is parallelizable (both on batch size and gradient computation) is desired for successful adversarial training. SDF possesses these crucial properties, making it a promising candidate for building more robust models.

We adversarially train a WRN-28-10 on CIFAR10. Similar to the procedure followed in Rony et al. (2019), we restrict $\ell_2$-norms of perturbation to 2.6 and set the maximum number of iterations for SDF to 6. We train the model on clean examples for the first 200 epochs, and we then fine-tune it with SDF generated adversarial examples for 60 more epochs. Since a model trained using DDN-generated samples Rony et al. (2019) has demonstrated greater robustness compared to a model trained using PGD Madry et al. (2017), we compare our model against the former. Our model reaches a test accuracy of $90.8\%$ while the model by Rony et al. (2019) obtains $89.0\%$. SDF adversarially trained model does not overfit to SDF attack because, as Table 6 shows, SDF obtains the smallest perturbation. It is evident that SDF adversarially trained model can significantly improve the robustness of model against minimum-norm attacks up to $30\%$. In terms of comparison of these two adversarially trained models with AutoAttack (AA), our model outperformed the Rony et al. (2019) by improving about $8.4\%$ against $\ell_\infty$-AA, for $\varepsilon = 8/255$, and $0.6\%$ against $\ell_2$-AA, for $\varepsilon = 0.5$.

Furthermore, compared to a network trained on DDN samples, our adversarially trained model has a smaller input curvature (Table 7). This observation corroborates the idea that a more robust network will exhibit a smaller input curvature Moosavi-Dezfooli et al. (2019); Srinivas et al.; Qin et al. (2019).

### 4.4 AutoAttack++

Table 8: Analysis of robust accuracy for various defense strategies against AA++ and AA with $\varepsilon = 0.5$ for six adversarially trained models on CIFAR-10. All models are taken from the RobustBench library Croce et al. (2020).

| Models | | AA | | AA++ | |
|---|---|---|---|---|---|
| | Clean acc. | Robust acc. | Grads | Robust acc. | Grads |
| R1 Rebuffi et al. (2021) | $95.7\%$ | $82.3\%$ | $1259.2$ | $\mathbf{82.1}\%$ | $\mathbf{599.5}$ |
| R2 Sehwag et al. (2021) | $90.3\%$ | $76.1\%$ | $1469.1$ | $76.1\%$ | $\mathbf{667.7}$ |
| R3 Gowal et al. (2020) | $89.4\%$ | $63.4\%$ | $1240.4$ | $\mathbf{62.2}\%$ | $\mathbf{431.5}$ |
| R4 Rice et al. (2020) | $88.6\%$ | $\mathbf{67.6}\%$ | $933.7$ | $68.4\%$ | $\mathbf{715.3}$ |
| R5 Rice et al. (2020) | $89.05\%$ | $66.4\%$ | $846.3$ | $\mathbf{62.5}\%$ | $\mathbf{613.7}$ |
| R6 Ding et al. (2018) | $88.02\%$ | $67.6\%$ | $721.4$ | $\mathbf{63.4}\%$ | $\mathbf{511.1}$ |
| Standard trained | $94.7\%$ | $0.00\%$ | $208.6$ | $0.00$ | $\mathbf{121.1}$ |

In this part, we introduce a new variant of AutoAttack by introducing AutoAttack++ (AA++). AutoAttack (AA) is a reliable and powerful ensemble attack that contains three types of white-box and a strong black-box attacks. AA evaluates the robustness of a trained model to adversarial perturbations whose $\ell_2/\ell_\infty$-norm is bounded by $\varepsilon$. By substituting SDF with the attacks in the AA, we significantly increase the performance of AA in terms of ***computational time***. Since SDF is an $\ell_2$-norm attack, we use the $\ell_2$-norm version of AA as well. We restrict maximum iterations of SDF to 10. If the norm of perturbations exceeds $\varepsilon$, we renormalize the perturbation to ensure its norm stays $\leq \varepsilon$. In this context, we have modified the AA algorithm by replacing APGD$^\top$ Croce & Hein (2020b) with SDF due to the former's cost and computation bottleneck in the context of AA. We compare the fooling rate and computational time of AA++ and AA on the stat-of-the-art models from the RobustBench leaderboard. In Table 8, we observe that AA++ is up to three times faster than AA. In an alternative scenario, we added the SDF to the beginning of the AA set, resulting in a version that is up to two times faster than the original AA, despite now containing five attacks (see Appendix). This outcome highlights the efficacy of SDF in finding adversarial examples. These experiments suggest that leveraging efficient ***minimum-norm*** and ***non-fixed iteration*** attacks, such as SDF, can enable faster and more reliable evaluation of the robustness of deep models.

## 5 Conclusion

In this work, we have introduced a family of parameter-free, fast, and parallelizable algorithms for crafting optimal adversarial perturbations. Our proposed algorithm, SDF, outperforms state-of-the-art $\ell_2$-norm attacks, while maintaining a small computational cost. We have demonstrated its effectiveness in various scenarios. Furthermore, we have shown that adversarial training using the examples generated by SDF builds more robust models. While our primary focus in this work has been on minimal $\ell_2$ attacks, there exists potential for extending SDF families to other threat models, including general $\ell_p$-norms and targeted attacks. In the Appendix, we have demonstrated straightforward modifications that highlight the applicability of SDF to both targeted and $\ell_\infty$-norm attacks. However, a more comprehensive evaluation remains a direction for future work. Moreover, further limitations of our proposed method are elaborated upon in Appendix O.

## REFERENCES

Maximilian Augustin, Alexander Meinke, and Matthias Hein. Adversarial robustness on in-and out-distribution improves explainability. In *European Conference on Computer Vision*, pp. 228–245. Springer, 2020.

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

Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*, 2019.

Jianbo Chen, Michael I Jordan, and Martin J Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *2020 ieee symposium on security and privacy (sp)*, pp. 1277–1294. IEEE, 2020.

Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*, pp. 2196–2205. PMLR, 2020a.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pp. 2206–2216. PMLR, 2020b.

Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020.

Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Advances in neural information processing systems*, 27, 2014.

Gavin Weiguang Ding, Yash Sharma, Kry Yik Chau Lui, and Ruitong Huang. Mma training: Direct input space margin maximization through adversarial training. *arXiv preprint arXiv:1812.02637*, 2018.

Logan Engstrom, Andrew Ilyas, Hadi Salman, Shibani Santurkar, and Dimitris Tsipras. Robustness (python library), 2019. URL `https://github.com/MadryLab/robustness`.

Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. The robustness of deep networks: A geometrical perspective. *IEEE Signal Processing Magazine*, 34(6):50–62, 2017.

Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, Pascal Frossard, and Stefano Soatto. Empirical study of the topology and geometry of deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3762–3770, 2018.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv preprint arXiv:2010.03593*, 2020.

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

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

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

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

Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pp. 319–345. Springer, 1999.

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

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černockỳ. Strategies for training large scale neural network language models. In *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*, pp. 196–201. IEEE, 2011.

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 (CVPR)*, June 2016.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Jonathan Uesato, and Pascal Frossard. Robustness via curvature regularization, and vice versa. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9078–9086, 2019.

Maura Pintor, Fabio Roli, Wieland Brendel, and Battista Biggio. Fast minimum-norm adversarial attacks through adaptive norm constraints. *Advances in Neural Information Processing Systems*, 34:20052–20062, 2021.

Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. *Advances in neural information processing systems*, 29, 2016.

Chongli Qin, James Martens, Sven Gowal, Dilip Krishnan, Krishnamurthy Dvijotham, Alhussein Fawzi, Soham De, Robert Stanforth, and Pushmeet Kohli. Adversarial robustness through local linearization. *Advances in Neural Information Processing Systems*, 32, 2019.

Rahul Rade and Seyed-Mohsen Moosavi-Dezfooli. Helper-based adversarial training: Reducing excessive margin to achieve a better accuracy vs. robustness trade-off. In *ICML 2021 Workshop on Adversarial Machine Learning*, 2021.

Ali Rahmati, Seyed-Mohsen Moosavi-Dezfooli, Pascal Frossard, and Huaiyu Dai. Geoda: a geometric framework for black-box adversarial attacks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8446–8455, 2020.

Sylvestre-Alvise Rebuffi, Sven Gowal, Dan A Calian, Florian Stimberg, Olivia Wiles, and Timothy Mann. Fixing data augmentation to improve adversarial robustness. *arXiv preprint arXiv:2103.01946*, 2021.

Leslie Rice, Eric Wong, and Zico Kolter. Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning*, pp. 8093–8104. PMLR, 2020.

Jerome Rony, Luiz G. Hafemann, Luiz S. Oliveira, Ismail Ben Ayed, Robert Sabourin, and Eric Granger. Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Jérôme Rony, Eric Granger, Marco Pedersoli, and Ismail Ben Ayed. Augmented lagrangian adversarial attacks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7738–7747, 2021.

Vikash Sehwag, Saeed Mahloujifar, Tinashe Handina, Sihui Dai, Chong Xiang, Mung Chiang, and Prateek Mittal. Improving adversarial robustness using proxy distributions. *CoRR*, abs/2104.09425, 2021.

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

Suraj Srinivas, Kyle Matoba, Himabindu Lakkaraju, and François Fleuret. Efficient training of low-curvature neural networks. In *Advances in Neural Information Processing Systems*.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. In *Advances in Neural Information Processing Systems*, 2020.

Jonathan Uesato, Brendan O'donoghue, Pushmeet Kohli, and Aaron Oord. Adversarial risk and the dangers of evaluating against weak attacks. In *International Conference on Machine Learning*, pp. 5025–5034. PMLR, 2018.

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

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. *arXiv preprint arXiv:1906.06316*, 2019.

## A PROOFS

**Proof of Proposition 1.**

Since $\nabla f(\boldsymbol{x})$ is Lipschitz-continuous, for $\boldsymbol{x}, \boldsymbol{y} \in B(\boldsymbol{x}_0, \epsilon)$, we have:

$$|f(\boldsymbol{x}) - f(\boldsymbol{y}) + \nabla f(\boldsymbol{y})^T(\boldsymbol{x} - \boldsymbol{y})| \leq \frac{L^{'}}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2 \tag{5}$$

DeepFool updates the new $\boldsymbol{x}_n$ in each according to the following equation:

$$\boldsymbol{x}_n = \boldsymbol{x}_{n-1} + \frac{\nabla f(\boldsymbol{x}_{n-1})}{\|\nabla f(\boldsymbol{x}_{n-1})\|_2} f(\boldsymbol{x}_{n-1}) \tag{6}$$

Hence if we substitute $\boldsymbol{x} = \boldsymbol{x}_n$ and $\boldsymbol{y} = \boldsymbol{x}_{n-1}$ in 5, we get:

$$|f(\boldsymbol{x}_n)| \leq \frac{L^{'}}{2}\|\boldsymbol{x}_n - \boldsymbol{x}_{n-1}\|^2. \tag{7}$$

Now, let $s_n := ||\boldsymbol{x}_n - \boldsymbol{x}_{n-1}||$. Using 7 and DeepFool's step, we get:

$$s_{n+1} = \frac{f(\boldsymbol{x}_n)}{\|\nabla f(\boldsymbol{x}_n)\|} \leq \frac{L^{'}}{2\zeta}\frac{f(\boldsymbol{x}_n)^2}{\|\nabla f(\boldsymbol{x}_n)\|^2} \tag{8}$$

We also know that for $\boldsymbol{x}, \boldsymbol{x}^* \in B(\boldsymbol{x}_0, \epsilon)$ and the Lipschitz property:

$$|f(\boldsymbol{x}) - f(\boldsymbol{x}^*)| \leq 2L^{'}\epsilon \tag{9}$$

From property $\boldsymbol{x}^*$ we know that $f(\boldsymbol{x}^*) = 0$, so:

$$|f(\boldsymbol{x})| \leq 2L^{'}\epsilon \tag{10}$$

$$s_{n+1} = \frac{f(\boldsymbol{x}_n)}{||\nabla f(\boldsymbol{x}_n)||} \leqslant s_n \epsilon \frac{L^{'2}}{\zeta^2} \tag{11}$$

Using the assumptions of the theorem, We have $\frac{L^{'}\epsilon}{\zeta^2} < 1$, and hence $s_n$ converges to 0 when $n \to \infty$. We conclude that $\{\boldsymbol{x}_n\}$ is a Cauchy sequence. Denote by $\boldsymbol{x}_\infty$ the limit point of $\{\boldsymbol{x}_n\}$. Using the continuity of $f$ and Eq.(7), we obtain

$$\lim_{n\to\infty} |f(\boldsymbol{x}_n)| = |f(\boldsymbol{x}_\infty)| = |f(\boldsymbol{x}^*)| = 0, \tag{12}$$

It is clear that $\boldsymbol{x}^*$ is $\boldsymbol{x}$. which concludes the proof of the theorem.

**Proof of Proposition 2.** Let us denote the acute angle between $\nabla f(\boldsymbol{x}_0 + \boldsymbol{r}_i)$ and $\boldsymbol{r}_i$ by $\theta_i$ ($0 \leq \theta_i \leq \pi/2$). Then from (4) we have $|\boldsymbol{r}_{i+1}| = |\boldsymbol{r}_i| \cos\theta_i$. Therefore, we get

$$|\boldsymbol{r}_{i+1}| = \prod_{i=1}^{i} \cos\theta_i |\boldsymbol{r}_0|. \tag{13}$$

Now there are two cases, either $\theta_i \to 0$ or not. Let us first consider the case where zero is not the limit of $\theta_i$. Then there exists some $\epsilon_0 > 0$ such that for any integer $N$ there exists some $n > N$ for which we have $\theta_n > \epsilon_0$. Now for $\epsilon_0$, we can have a series of integers $n_i$ where for all of them we have $\theta_{n_i} > \epsilon_0$. Since we have $0 \leq |\cos\theta| \leq 1$, we have the following inequality:

$$0 \leq \prod_{i=0}^{\infty} |\cos\theta_i| \leq \prod_{i=0}^{\infty} |\cos\theta_{n_i}| \leq \prod_{i=0}^{\infty} |\cos\epsilon_0| \tag{14}$$

The RHS of the above inequality goes to zero which proves that $\boldsymbol{r}_i \to 0$. This leaves us with the other case where $\theta_i \to 0$. This means that $\cos\theta_i \to 1$ which is the maximum of Eq. (3), this completes the proof.

Table 9: Comparison of the effectiveness of line search on the CIFAR10 data for SDF and DF. We use one regularly trained model S (WRN-28-10) and three adversarially trained models (shown with R1 Rony et al. (2019), R2 Augustin et al. (2020) and R3 Rade & Moosavi-Dezfooli (2021)). ✓and ✗ indicate the presence and absence of line search respectively.

| Model | DF | | SDF | |
|---|---|---|---|---|
| | ✓ | ✗ | ✓ | ✗ |
| S | 0.16 | 0.19 | **0.09** | 0.10 |
| R1 | 0.87 | 1.02 | **0.73** | 0.76 |
| R2 | 1.40 | 1.73 | **0.91** | 0.93 |
| R3 | 1.13 | 1.36 | **1.04** | 1.09 |

Table 10: Comparison of the effectiveness of line search on the CIFAR-10 data for other attacks. Line search effects are a little for DDN and ALMA. For FMN and FAB because they use line search at the end of their algorithms (they remind this algorithm as a *binary search* and *final search*, respectively), line search does not become effective.

| Model | DDN | | ALMA | | FMN | | FAB | |
|---|---|---|---|---|---|---|---|---|
| | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ |
| WRN-28-10 | 0.12 | 0.13 | 0.10 | 0.10 | 0.11 | 0.11 | 0.11 | 0.11 |
| R1 Rony et al. (2019) | 0.73 | 0.73 | 0.71 | 0.71 | 1.10 | 1.10 | 0.75 | 0.75 |
| R2 Augustin et al. (2020) | 0.96 | 0.97 | 0.93 | 0.94 | 0.95 | 0.95 | 1.03 | 1.03 |
| R3 Rade & Moosavi-Dezfooli (2021) | 1.04 | 1.04 | 1.06 | 1.06 | 1.08 | 1.08 | 1.07 | 1.07 |

## B On the benefits of line search

As we show in Figure 2 DF typically finds an overly perturbed point. SDF's gradients depend on DF, so overly perturbing DF is problematic. Line search is a mechanism that we add to the end of our algorithms to tackle this problem. For a fair comparison between adversarial attacks, we add this algorithm to the end of other algorithms to investigate the effectiveness of line search.

As shown in Table 9, we observe that line search can increase the performance of the DF significantly. However, this effectiveness for SDF is a little.

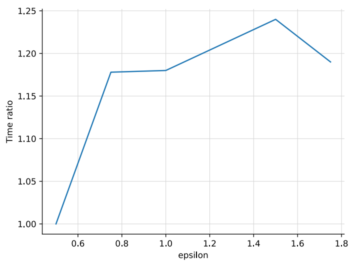
We now measure the effectiveness of line search for other attacks. As observed from Table 10, line search effectiveness for DDN and ALMA is small.

## C Comparison on CIFAR10 with the PRN-18

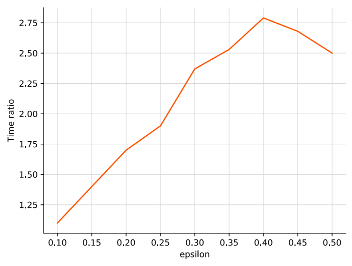
In this section, we compare SDF with other minimum-norm attacks against an adversarially trained network Rade & Moosavi-Dezfooli (2021). In Table 11, SDF achieves smaller perturbation compared to other attacks, whereas it costs only half as much as other attacks.

Table 11: Comparison of SDF with other state-of-the-art attacks for median $\ell_2$ on CIFAR-10 dataset for adversarially trained network (PRN-18 Rade & Moosavi-Dezfooli (2021)).

| Attack | FR | Median-$\ell_2$ | Grads |
|---|---|---|---|
| ALMA | 100 | 0.68 | 100 |
| DDN | 100 | 0.73 | 100 |
| FAB | 100 | 0.77 | 210 |
| FMN | 99.7 | 0.81 | 100 |
| SDF | 100 | **0.65** | **46** |



(a) R1



(b) S

Figure 4: In this figure, we show the time ratio of AA to AA++. For regularly trained model (WRN-28-10) and adversarially trained model Rade & Moosavi-Dezfooli (2021) (R1). We perform this experiment on 1000 samples from CIFAR10 data.

## D Performance comparison of adversarially trained models versus Auto-Attack (AA)

Evaluating the adversarially trained models with attacks used in the training process is not a standard evaluation in the robustness literature. For this reason, we evaluate robust models with AA. We perform this experiment with two modes; first, we measure the robustness of models with $\ell_\infty$ norm, and in a second mode, we evaluate them in terms of $\ell_2$ norm. Tables 12 and 13 show that adversarial training with SDF samples is more robust against reliable AA than the model trained on DDN samples Rony et al. (2019).

Table 12: Robustness results of adversarially trained models on CIFAR-10 with $\ell_\infty$-AA. We perform this experiment on 1000 samples for each $\epsilon$.

| Model | Natural | $\varepsilon = \frac{6}{255}$ | $\frac{8}{255}$ | $\frac{10}{255}$ |
|---|---|---|---|---|
| DDN | 89.1 | 45 | 29.6 | 17.6 |
| SDF (Ours) | 90.8 | **47.5** | **38.1** | **25.4** |

Table 13: Robustness results of adversarially trained models on CIFAR-10 with $\ell_2$-AA. We perform this experiment on 1000 samples for each $\epsilon$.

| Model | Natural | $\varepsilon = 0.3$ | 0.4 | 0.5 | 0.6 |
|---|---|---|---|---|---|
| DDN | 89.1 | 78.1 | 73 | 67.5 | 61.7 |
| SDF (Ours) | 90.8 | **83.1** | **79.7** | **68.1** | **63.9** |

## E Another variants of AA++

As we mentioned, in an alternative scenario, we added the SDF to the beginning of the AA set, resulting in a version that is up to two times faster than the original AA. In this scenario, we do not exchange the SDF with APGD. We add SDF to the AA configuration. So in this configuration, AA has five attacks (SDF, APGD, APGD$^\top$, FAB, Square). By this design, we guarantee the performance of AA. An interesting phenomenon observed from these tables is that when the budget increases, the speed of the AA++ increases. We should note that we restrict the number of iterations for SDF to 10.

## F Why do we need stronger minimum-norm attacks?

Bounded-norm attacks like FGSM Goodfellow et al. (2014), PGD Madry et al. (2017), and momentum variants of PGD Uesato et al. (2018), by optimizing the difference between the logits of the true class and the best non-true class, try to find an adversarial region with maximum confidence within a given, fixed perturbation size. Bounded-norm attacks only evaluate the robustness of deep neural networks; this means that they report a single scalar value as robust accuracy for a fixed budget. The

Table 14: This table shows the $\ell_2$-median for the minimum-norm attacks. For all networks, we set learning rate = $0.01$ and weight decay = $0.01$. For training with Lp-pooling, we set $p = 2$ for all settings.

| ATTACK | RN18 | | | MOBILENET | | |
|---|---|---|---|---|---|---|
| | NO POOL | MAX-POOL | LP-POOL | NO POOL | MAX-POOL | LP-POOL |
| DF | 0.40 | 0.90 | 0.91 | 0.51 | 0.95 | 0.93 |
| DDN | 0.16 | 0.25 | 0.26 | 0.22 | 0.27 | 0.26 |
| FMN | 0.18 | 0.27 | 0.30 | 0.24 | 0.30 | 0.29 |
| C&W | 0.18 | 0.25 | 0.27 | 0.22 | 0.26 | 0.24 |
| ALMA | 0.19 | 0.23 | 0.23 | **0.20** | 0.25 | 0.22 |
| SDF | **0.16** | **0.21** | **0.22** | **0.20** | **0.23** | **0.21** |

superiority of minimum-norm attacks is to report a distribution of perturbation norms, and they do not report a percentage of fooling rates (robust accuracy) by a single scalar value. This critical property of minimum-norm attacks helps to accelerate to take an in-depth intuition about the geometrical behavior of deep neural networks.

We aim to address a phenomenon we observe by using the superiority of minimum-norm attacks. We observed that a minor change within the design of deep neural networks affects the performance of adversarial attacks. To show the superiority of minimum-norm attacks, we show how minimum-norm attacks verify these minor changes rather than bounded-norm attacks.

Modeling with max-pooling was a fundamental aspect of convolutional neural networks when they were first introduced as the best image classifiers. Some state-of-the-art classifiers such as Krizhevsky et al. (2017); Simonyan & Zisserman (2014); He et al. (2016a) use this layer in network configuration. We use the pooling layers to show that using the max-pooling and Lp-pooling layer in the network design leads to finding perturbation with a bigger $\ell_2$-norm.

Assume that we have a classifier $f$. We train $f$ in two modes until the training loss converges. In the first mode, $f$ is trained in the presence of the pooling layer in its configuration, and in the second mode, $f$ does not have a pooling layer. When we measure the robustness of these two networks with regular budgets used in bounded-norms attacks like PGD ($\varepsilon = 8/255$), we observe that the robust accuracy is equal to $0\%$. This is precisely where bounded-norm attacks such as PGD mislead robustness literature in its assumptions regarding deep neural network properties. However, a solution to solve the problem of bounded-norm attack scan be proposed: *" Analyzing the quantity of changes in robust accuracy across different epsilons reveal these minor changes."* Is this case, the solution is costly. This is precisely where the distributive view of perturbations from worst-case to best-case of minimum-norm attacks detects this minor change.

To show these changes, we trained ResNet-18 and Mobile-Net Howard et al. (2017) in two settings. In the first setting, we trained them in the presence of a pooling layer until the training loss converged, and in the second setting, we trained them in the absence of a pooling layer until the training loss converged. We should note that we remove all pooling-layers in these two settings. For a fair comparison, we train models until they achieve zero training loss using a multi-step learning rate. We use max-pooling and Lp-pooling, for $p = 2$, for this minor changes.

Table 14 shows that using a pooling layer in network configuration can increase robustness. DF has an entirely different behavior according to the presence or absence of the pooling layer; max-pooling affects up to $50\%$ of DF performance. This effect is up to $9\%$ for DDN and FMN. ALMA and SDF show a $4\%$ impact in their performance, which shows their consistency compared to other attacks.

As shown in Table 15, we observe that models with pooling-layers have more robust accuracy when facing adversarial attacks such as AA and PGD. It should be noted that using regular epsilon for AA and PGD will not demonstrate these modifications. For this reason, we choose an epsilon for AA and PGD lower ($\varepsilon = 2/255$) than the regular format ($\varepsilon = 8/255$).

Table 14 and 15 demonstrate that pooling-layers can affect adversarial robustness of deep networks. Powerful attacks such as SDF and ALMA show high consistency in these setups, highlighting the need for powerful attacks.

Table 15: This table shows the robust accuracy for all networks against to the AA and PGD. For training with Lp-pooling, we set $p = 2$ for all settings.

| ATTACK | RN18 | | | MOBILENET | | |
|---|---|---|---|---|---|---|
| | NO POOL | MAX-POOL | LP-POOL | NO POOL | MAX-POOL | LP-POOL |
| AA | 1.1% | 17.2% | 16.3% | 8.7% | 21.3% | 20.2% |
| PGD | 9.3% | 28% | 26.2% | 16.8% | 31.4% | 28.7% |

### F.1 MAX-POOLING'S EFFECT ON THE DECISION BOUNDARY'S CURVATURE

Here, we take a step further and investigate why max-pooling impacts the robustness of models. In order to perform this analysis, we analyze gradient norms, Hessian norms, and the model's curvature. The curvature of a point is a mathematical quantity that indicates the degree of non-linearity. It has been observed that robust models are characterized by their small curvature Moosavi-Dezfooli et al. (2019), implying smaller Hessian norms. In order to investigate robustness independent of non-linearity, Srinivas et al. propose *normalized curvature*, which normalizes the Hessian norm at a given input $\boldsymbol{x}$ by its corresponding gradient norm. They defined *normalized curvature* for a neural network classifier $f$ as $\mathcal{C}_f(\boldsymbol{x}) = \|\nabla^2 f(\boldsymbol{x})\|_2/(\|\nabla f(\boldsymbol{x})\|_2 + \varepsilon)$. Where $\|\nabla f(\boldsymbol{x})\|_2$ and $\|\nabla^2 f(\boldsymbol{x})\|_2$ are the $\ell_2$-norm of the gradient and the spectral norm of the Hessian, respectively, where $\nabla f(\boldsymbol{x}) \in \mathbb{R}^d$, $\nabla^2 f(\boldsymbol{x}) \in \mathbb{R}^{d\times d}$, and $\varepsilon > 0$ is a small constant to ensure the proper behavior of the measure. In Table 16, we measure these quantities for two trained models, one with max-pooling and one without. It clearly shows that the model incorporating max-pooling exhibits a smaller curvature. This finding corroborates the observation that models with greater robustness tend to have a smaller curvature value.

Table 16: Model geometry of different ResNet-18 models. W (with pooling) and W/O (without pooling).

| MODEL | $\mathbb{E}_{\boldsymbol{x}}\|\nabla f(\boldsymbol{x})\|_2$ | $\mathbb{E}_{\boldsymbol{x}}\|\nabla^2 f(\boldsymbol{x})\|_2$ | $\mathbb{E}_{\boldsymbol{x}}\mathcal{C}_f(\boldsymbol{x})$ |
|---|---|---|---|
| W | **4.75** ± 1.54 | **120.70** ± 48.74 | **14.94** ± 0.52 |
| W/O | 7.04 ± 2.44 | 269.74 ± 10.23 | 22.81 ± 2.58 |

## G MODEL GEOMETRY FOR AT MODELS

In this section we provide curvature analysis of our adversarially trained networks, SDF AT, and DDN AT model. Table 17 shows that our AT model decreases the curvature of network more than DDN AT model.

## H ORTHOGONALITY CONDITIONS FOR OTHER ATTACKS

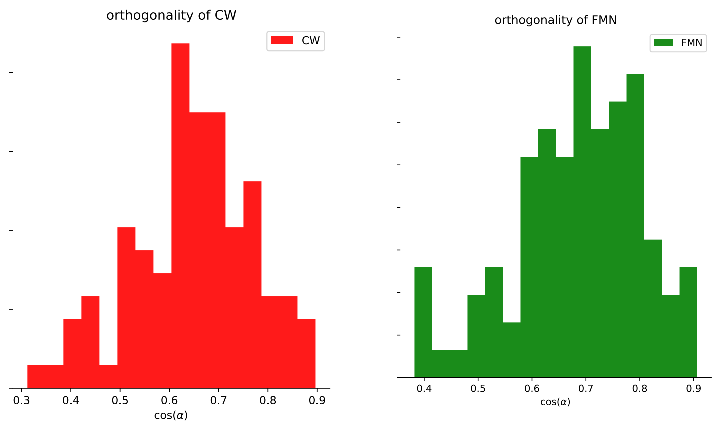
In Fig 5, we show the histogram of the cosine angle distribution between normals to the decision boundary and perturbations for C&W and FMN attacks.

## I CNN ARCHITECTURE USED IN TABLE 2

The architecture used to compare SDF variants and DF (Table 2) is summarized in Table 18.

Table 17: Model geometry for regular and adversarially trained models.

| MODEL | $\mathbb{E}_{\boldsymbol{x}}\|\nabla f(\boldsymbol{x})\|_2$ | $\mathbb{E}_{\boldsymbol{x}}\|\nabla^2 f(\boldsymbol{x})\|_2$ | $\mathbb{E}_{\boldsymbol{x}}\mathcal{C}_f(\boldsymbol{x})$ |
|---|---|---|---|
| STANDARD | 9.54 ± 1.02 | 600.06 ± 29.76 | 73.99 ± 6.62 |
| DDN AT | 0.91 ± 0.34 | 2.86 ± 1.22 | 4.32 ± 2.91 |
| SDF AT | **0.38** ± 0.60 | **0.73** ± 0.08 | **1.66** ± 0.86 |



C&W FMN

Figure 5: Histogram of the cosine angle distribution between the gradient in the last step of C&W( ***left***), FMN( ***right***) and the perturbation vector obtained by C&W and FMN. As previously mentioned in the analysis of *Linesearch* B, it is observed that a majority of adversarial attacks exhibit not-overly perturbed perturbations due to the incorporation of the *Linesearch* mechanism into their algorithms. Therefore, due to this justification, gamma analysis is not an effective tool for distinguishing power and comparing algorithms.

| Layer Type | CIFAR-10 |
|---|---|
| Convolution + ReLU | $3 \times 3 \times 64$ |
| Convolution + ReLU | $3 \times 3 \times 64$ |
| max-pooling | $2 \times 2$ |
| Convolution + ReLU | $3 \times 3 \times 128$ |
| Convolution + ReLU | $3 \times 3 \times 128$ |
| max-pooling | $2 \times 2$ |
| Fully Connected + ReLU | 256 |
| Fully Connected + ReLU | 256 |
| Fully Connected + Softmax | 10 |

Table 18: CNN architecture.

## J MULTI-CLASS ALGORITHMS FOR SDF$(1, 3)$ AND SDF$(1, 1)$

Algorithm 3 and 4 summarizes pseudo-codes for the multi-class versions of SDF$(1, 1)$ and SDF$(1, 3)$.

**Algorithm 3:** SDF (1,1)

```
Input: image $\boldsymbol{x}$, classifier $f$.
Output: perturbation $\boldsymbol{r}$
1  Initialize: $\boldsymbol{x}_0 \leftarrow \boldsymbol{x},\ \ i \leftarrow 0$
2  while $\hat{k}(\boldsymbol{x}_i) = \hat{k}(\boldsymbol{x}_0)$ do
3      for $k \neq \hat{k}(\boldsymbol{x}_0)$ do
4          $\boldsymbol{w}'_k \leftarrow \nabla f_k(\boldsymbol{x}_i) - \nabla f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
5          $f'_k \leftarrow f_k(\boldsymbol{x}_i) - f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
6      end
7      $\hat{l} \leftarrow \arg\min_{k \neq \hat{k}(\boldsymbol{x}_0)} \frac{|f'_k|}{\|\boldsymbol{w}'_k\|_2}$
8      $\widetilde{\boldsymbol{r}} \leftarrow \frac{|f'_{\hat{l}}|}{\|\boldsymbol{w}'_{\hat{l}}\|_2^2} \boldsymbol{w}'_{\hat{l}}$
9      $\widetilde{\boldsymbol{x}}_i = \boldsymbol{x}_i + \widetilde{\boldsymbol{r}}$
10     $\boldsymbol{w}_i \leftarrow \nabla f_{k(\widetilde{\boldsymbol{x}}_i)}(\widetilde{\boldsymbol{x}}_i) - \nabla f_{k(\boldsymbol{x}_0)}(\widetilde{\boldsymbol{x}}_i)$
11     $\boldsymbol{x} \leftarrow \boldsymbol{x}_0 + \frac{(\widetilde{\boldsymbol{x}}_i - \boldsymbol{x}_0)^\top \boldsymbol{w}_i}{\|\boldsymbol{w}_i\|^2} \boldsymbol{w}_i$
12     $i \leftarrow i + 1$
13 end
14 return $\boldsymbol{r} = \boldsymbol{x}_i - \boldsymbol{x}_0$
```

**Algorithm 4:** SDF (1,3)

```
Input: image $\boldsymbol{x}$, classifier $f$.
Output: perturbation $\boldsymbol{r}$
1  Initialize: $\boldsymbol{x}_0 \leftarrow \boldsymbol{x},\ \ i \leftarrow 0$
2  while $\hat{k}(\boldsymbol{x}_i) = \hat{k}(\boldsymbol{x}_0)$ do
3      for $k \neq \hat{k}(\boldsymbol{x}_0)$ do
4          $\boldsymbol{w}'_k \leftarrow \nabla f_k(\boldsymbol{x}_i) - \nabla f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
5          $f'_k \leftarrow f_k(\boldsymbol{x}_i) - f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
6      end
7      $\hat{l} \leftarrow \arg\min_{k \neq \hat{k}(\boldsymbol{x}_0)} \frac{|f'_k|}{\|\boldsymbol{w}'_k\|_2}$
8      $\widetilde{\boldsymbol{r}} \leftarrow \frac{|f'_{\hat{l}}|}{\|\boldsymbol{w}'_{\hat{l}}\|_2^2} \boldsymbol{w}'_{\hat{l}}$
9      $\widetilde{\boldsymbol{x}}_i = \boldsymbol{x}_i + \widetilde{\boldsymbol{r}}$
10     for 3 *steps* do
11         $\boldsymbol{w}_i \leftarrow \nabla f_{k(\widetilde{\boldsymbol{x}}_i)}(\widetilde{\boldsymbol{x}}_i) - \nabla f_{k(\boldsymbol{x}_0)}(\widetilde{\boldsymbol{x}}_i)$
12         $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_0 + \frac{(\widetilde{\boldsymbol{x}}_i - \boldsymbol{x}_0)^\top \boldsymbol{w}_i}{\|\boldsymbol{w}_i\|^2} \boldsymbol{w}_i$
13     end
14     $i \leftarrow i + 1$
15 end
16 return $\boldsymbol{r} = \boldsymbol{x}_i - \boldsymbol{x}_0$
```

## K Natural trained MNIST model

In Table 19 we show the results of evaluating adversarial attacks on naturally trained SmallCNN on MNIST dataset. Our algorithm demonstrates a higher rate of convergence compared to other algorithms, as the perturbations for all algorithms are generally similar.

Table 19: We compare the performance of all algorithms on the natural SmallCNN model that was trained on the MNIST dataset.

| Attacks | FR | Median-$\ell_2$ | Grads |
|---|---|---|---|
| ALMA | 100 | **1.34** | 1000 |
| DDN | 100 | 1.36 | 1000 |
| FAB | 100 | 1.36 | 10000 |
| FMN | 97.10 | 1.37 | 1000 |
| C&W | 99.80 | 1.35 | 90000 |
| SDF | 100 | **1.34** | **67** |

## L Runtime comparison

We report the number of gradient computations as a main proxy for computional cost comparison. In Table 20, we have compared the runtime of different attacks for a fixed hardware. SDF is significantly faster.

Table 20: Runtime comparison for adversarial attacks on WRN-28-10 architecture trained on CIFAR10, for both naturally trained model and adversarially trained models.

| | Natural | | R1 Rebuffi et al. (2021) | |
|---|---|---|---|---|
| Attacks | Time (S) | Median-$\ell_2$ | Time (S) | Median-$\ell_2$ |
| ALMA | 1.71 | 0.10 | 13.10 | 1.22 |
| DDN | 1.54 | 0.13 | 12.44 | 1.53 |
| FAB | 2.33 | 0.11 | 16.21 | 1.66 |
| FMN | 1.42 | 0.11 | 10.25 | 1.83 |
| C&W | 734.8 | 0.12 | 5402.1 | 1.68 |
| SDF | **0.48** | **0.09** | **2.93** | **1.19** |

## M Geometric conditions of optimal perturbations

Figure 6 illustrates the two conditions discussed in Section 2. In the figure, $n_1$ and $n_2$ represent two orthogonal vectors to the decision boundary. The optimal perturbation vector $\boldsymbol{r}^*$ aligns parallel to $n_2$. On the other hand, a non-optimal perturbation $\boldsymbol{r}_{\text{DF}}$ forms an angle $\alpha$ with $n_1$.

## N Query-distortion curves

As demonstrated in Pintor et al. (2021), query-distortion curves are utilised as a metric for evaluating computational complexity of white-box attacks. In this particular context, the term “query” refers to the quantity of forward passes available to find adversarial perturbations.

Unlike FMN and ALMA, SDF (and DF) does not allow control over the number of forward and backward computations. They typically stop once a successful adversarial example is found. Terminating the process prematurely could prevent them from finding an adversarial example. Hence, we instead opted to plot the median norm of achievable perturbations for a given maximum number of queries (Figure 7) Although this is not directly comparable to the query-distortion curves in Pintor et al. (2021), it provides a more comprehensive view of the query distribution than the median alone.

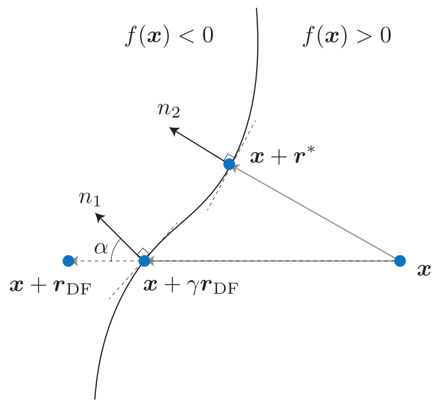

$f(\boldsymbol{x}) < 0$
$f(\boldsymbol{x}) > 0$
$n_2$
$\boldsymbol{x} + \boldsymbol{r}^*$
$n_1$
$\alpha$
$\boldsymbol{x}$
$\boldsymbol{x} + \boldsymbol{r}_{\mathrm{DF}}$
$\boldsymbol{x} + \gamma\boldsymbol{r}_{\mathrm{DF}}$

Figure 6: Illustration of the optimal adversarial example $\boldsymbol{x} + \boldsymbol{r}^*$ for a binary classifier $f$; the example lies on the decision boundary (set of points where $f(\boldsymbol{x}) = 0$) and the perturbation vector $\boldsymbol{r}^*$ is orthogonal to this boundary.

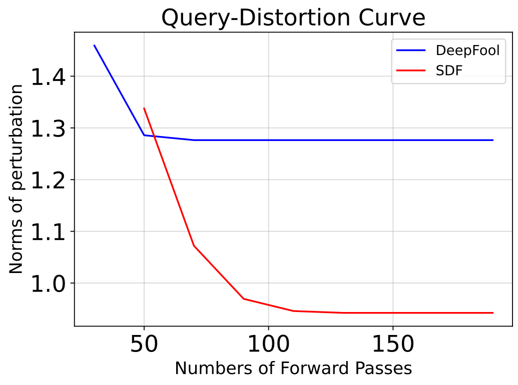


Figure 7: Query-perturbation curves illustrating the relationship between the number of queries and norms of perturbations for the DF and SDF attacks. The attacks are performed on a WRN-28-10 adversarially trained on CIFAR-10 dataset.

## O Limitations

In this section, we discuss some limitations and potential extensions of SDF.

**Extension to other $\ell_p$-norms and targeted attacks.** The proposed attack is primarily designed for $\ell_2$-norm adversarial perturbations. Moreover, our method, similar to DeepFool (DF), is non-targeted. Though there are potential approaches for adapting SDF to targeted and $\ell_p$ attacks, these aspects remain largely unexplored in our work.

Nevertheless, we here demonstrate how one could possibly extend SDF to other $p$-norms. A simple way is to replace the $\ell_2$ projection (Line 5 of Algorithm 2) with a projection operator minimizing $\ell_p$ norm similar to the derivations used in Moosavi-Dezfooli et al. (2016). In particular, for $p = \infty$, the following projection would replace the line 5 of Algorithm 2:

$$\boldsymbol{x} \leftarrow \boldsymbol{x}_0 + \frac{(\widetilde{\boldsymbol{x}} - \boldsymbol{x}_0)^\top \boldsymbol{w}}{||\boldsymbol{w}||_1} \operatorname{sign}(\boldsymbol{w}) \tag{15}$$

In Table 21, we compare the performance of this modified version of SDF, named SDF$_{\ell_\infty}$ with FMN, FAB, and DF, on two pretrained networks M1 Madry et al. (2017) and M2 Rony et al. (2019) on CIFAR-10 dataset. Our findings indicate that SDF$_{\ell_\infty}$ also exhibits superior performance compared to other algorithms in discovering smaller perturbations.

Table 21: Performance of SDF$_{\ell_\infty}$ on two robust networks trained on CIFAR-10 dataset.

| Attacks | M1 | | | M2 | | |
|---|---|---|---|---|---|---|
| | Median $\ell_\infty$ | FR | Grads | Median $\ell_\infty$ | FR | Grads |
| DF | 0.031 | 96.7 | **24** | 0.043 | 97.4 | **31** |
| FAB | 0.025 | 99.1 | 100 | 0.038 | 99.6 | 100 |
| FMN | 0.024 | **100** | 100 | 0.035 | **100** | 100 |
| SDF$_{\ell_\infty}$ | **0.019** | **100** | 33 | **0.027** | **100** | 46 |

Table 22: Performance of targeted SDF on a standard trained WRN-28-10 on CIFAR-10, measured using 100 random samples.

| Attacks | Targeted | | | | Untargeted | | | |
|---|---|---|---|---|---|---|---|---|
| | FR | Mean $\ell_2$ | Median $\ell_2$ | Grads | FR | Mean $\ell_2$ | Median $\ell_2$ | Grads |
| DDN | 100 | 0.24 | 0.25 | 100 | 100 | 0.13 | 0.14 | 100 |
| FMN | 96.2 | 0.22 | 0.24 | 100 | 97.3 | 0.11 | 0.13 | 100 |
| SDF (targeted) | 98.2 | 0.21 | 0.22 | 25 | 100 | 0.10 | 0.11 | 34 |

Furthermore, we can convert SDF to a targeted attack by replacing the line 3 of Algorithm 2 with the targeted version of DeepFool, and the line 4 with the following:

$$\boldsymbol{w} \leftarrow \nabla f_t(\widetilde{\boldsymbol{x}}) - \nabla f_{\hat{k}(\boldsymbol{x}_0)}(\widetilde{\boldsymbol{x}}), \tag{16}$$

where $t$ is the target label. We followed the procedure outlined in Carlini & Wagner (2017) to measure the performance in the targeted setting. The result is summarized in Table 22. While SDF is effective in quickly finding smaller perturbations, it does not achieve a 100% fooling rate. Further analysis is required to understand the factors preventing SDF from converging in certain cases. This aspect remains an area for future work.

**Convergence guarantees.** A common challenge for all gradient-based optimization methods applied to non-convex problems is the lack of a guarantee in finding globally optimal perturbations for SotA neural networks. Obtaining even local guarantees is not trivial. Nevertheless, in Propositions 1 and 2 we worked towards this goal. We have established local guarantees showing the convergence of each individual operation, namely the DeepFool step and projection step. However, further analysis is needed to establish local guarantees for the overall algorithm.

**Adaptive attacks.** It is known that gradient-based attacks, ours included, are prone to gradient obfuscation/masking Carlini et al. (2019). To counter this challenge, adaptation, as outlined in Tramer et al. (2020), is needed. It is also important to recognize that adapting geometric attacks such as SDF, does not follow a one-size-fits-all approach, as opposed to loss-based ones such as PGD. While this might be perceived as a weakness, it actually underscores a broader trend in the community. The predominant focus has been on loss-based attacks. This emphasis has inadvertently led to less exploration and development in the realm of geometric attacks.