SOAR: SECOND-ORDER ADVERSARIAL REGULARIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Adversarial training is a common approach to improving the robustness of deep neural networks against adversarial examples. In this work, we propose a novel regularization approach as an alternative. To derive the regularizer, we formulate the adversarial robustness problem under the robust optimization framework and approximate the loss function using a second-order Taylor series expansion. Our proposed *second-order adversarial regularizer* (SOAR) is an upper bound based on the Taylor approximation of the inner-max in the robust optimization objective. We empirically show that the proposed method improves the robustness of networks against the ℓ_∞ and ℓ_2 bounded perturbations on CIFAR-10 and SVHN.

1 INTRODUCTION

Adversarial training (Szegedy et al., 2013) is the standard approach for improving the robustness of deep neural networks (DNN), or any other model, against adversarial examples. It is a data augmentation method that adds adversarial examples to the training set and updates the network with newly added data points. Intuitively, this procedure encourages the DNN not to make the same mistakes against an adversary. By adding sufficiently enough adversarial examples, the network gradually becomes robust to the attack it was trained on. One of the challenges with such a data augmentation approach is the tremendous amount of additional data required for learning a robust model. Schmidt et al. (2018) show that under a Gaussian data model, the sample complexity of robust generalization is $\sqrt{d}$ times larger than that of standard generalization. They further suggest that current datasets (e.g., CIFAR-10) may not be large enough to attain higher adversarial accuracy. A data augmentation procedure, however, is an indirect way to improve the robustness of a DNN. Our proposed alternative is to define a regularizer that penalizes DNN parameters prone to attacks. Minimizing the regularized loss function leads to estimators robust to adversarial examples.

Adversarial training and our proposal can both be formulated in terms of robust optimization framework for adversarial robustness (Ben-Tal et al., 2009; Madry et al., 2018; Wong & Kolter, 2018; Shaham et al., 2018; Sinha et al., 2018). In this formulation, one is seeking to improve the *worst-case* performance of the model, where the performance is measured by a particular loss function ℓ. Adversarial training can be understood as approximating such a worst-case loss by finding the corresponding worst-case data point, i.e., $x + \delta$ with some specific attack techniques. Our proposed method is more direct. It is based on approximating the loss function $\ell(x + \delta)$ using its second-order Taylor series expansion, i.e.,

$$\ell(x+\delta) \approx \ell(x) + \nabla_x \ell(x)^\top \delta + \frac{1}{2}\delta^\top \nabla_x^2 \ell(x)\delta,$$

and then upper bounding the worst-case loss using the expansion terms. By considering both gradient and Hessian of the loss function with respect to (w.r.t.) the input, we can provide a more accurate approximation to the worst-case loss. In our derivations, we consider both ℓ_2 and ℓ_∞ attacks. In our derivations, the second-order expansion incorporates both the gradient and Hessian of the loss function with respect to (w.r.t.) the input. We call the method *Second-Order Adversarial Regularizer (SOAR)* (not to be confused with the Soar cognitive architecture Laird 2012). In the course of development of SOAR, we make the following contributions:

- We show that an over-parameterized linear regression model can be severely affected by an adversary, even though its population loss is zero. We robustify it with a regularizer that

exactly mimics the adversarial training. This suggests that regularization can be used instead of adversarial training (Section 2).

- Inspired by such a possibility, we develop a regularizer which upper bounds the **worst-case** effect of an adversary under an approximation of the loss. In particular, we derive SOAR, which approximates the inner maximization of the robust optimization formulation based on the second-order Taylor series expansion of the loss function (Section 4).
- We study SOAR in the logistic regression setting and reveal challenges with regularization using Hessian w.r.t. the input. We develop a simple initialization method to circumvent the issue (Section 4.1).
- We empirically show that SOAR significantly improves the adversarial robustness of the network against ℓ_∞ attacks ($\varepsilon = \frac{8}{255}$) and ℓ_2 attacks ($\varepsilon = 1.74$) on CIFAR-10 and SVHN. Specifically, we evaluate using a PGD1000 white-box attack, transferred PGD1000 attacks and SimBA (Guo et al., 2019). Its adversarial robustness is competitive to SOTA methods, while its negative effect on the standard accuracy (on clean data) is minimum (Section 5).

2 Linear Regression with an Over-parametrized Model

This section shows that for over-parameterized linear models, gradient descent (GD) finds a solution that has zero population loss, but is prone to attacks. It also shows that one can avoid this problem with defining an appropriate regularizer. Hence, we do not need adversarial training to robustify such a model. This simple illustration motivates the development of our method in next sections. We only briefly report the main results here, and defer the derivations to Appendix A .

Consider a linear model $f_w(x) = \langle\, w\,,\, x\,\rangle$ with $x, w \in \mathbb{R}^d$. Suppose that $w^* = (1, 0, 0, \ldots, 0)^\top$ and the distribution of $x \sim p$ is such that it is confined on a 1-dimensional subspace $\{\,(x_1, 0, 0, \ldots, 0) \,:\, x_1 \in \mathbb{R}\,\}$. This setup can be thought of as using an over-parameterized model that has many irrelevant dimensions with data that is only covering the relevant dimension of the input space. This is a simplified model of the situation when the data manifold has a dimension lower than the input space. We consider the squared error pointwise loss $l(x; w) = \frac{1}{2}\left|\langle\, x\,,\, w\,\rangle - \langle\, x\,,\, w^*\,\rangle\right|^2$. Denote the residual by $r(x; w) = \langle\, x\,,\, w - w^*\,\rangle$, and the population loss by $\mathcal{L}(w) = \mathbb{E}\left[l(X; w)\right]$.

Suppose that we initialize the weights as $w(0) = W \sim N(0, \sigma^2 \mathbf{I}_{d\times d})$, and use GD on the population loss, i.e., $w(t+1) \leftarrow w(t) - \beta \nabla_w \mathcal{L}(w)$. It is easy to see that the partial derivatives w.r.t. $w_{2,\ldots,d}$ are all zero, i.e., no weight adaptation happens. With a proper choice of learning rate β, we get that the asymptotic solution is $\bar{w} \triangleq \lim_{r\to\infty} w(t) = (w_1^*, w_2(0), w_3(0), \ldots, w_d(0))^\top$. That is, the initial random weights on dimensions $2, \ldots, d$ do not change.

We make two observations. The first is that $\mathcal{L}(\bar{w}) = 0$, i.e., the population loss is zero. So from the perspective of training under the original loss, we are finding the optimal solution. The second observation is that this model is vulnerable to adversarial examples. An FGSM-like attack that perturbs x by $\Delta x = (0, \Delta x_2, \Delta x_3, \ldots, \Delta x_d)^\top$ with $\Delta x_i = \varepsilon\, \mathrm{sign}(w_i(0))$ (for $i = 2, \ldots, d$) has the population loss of $\mathbb{E}_{X,W}\left[l(X + \Delta x); \bar{w})\right] \approx O(\varepsilon^2 d^2 \sigma^2)$ under the adversary at the asymptotic solution $\bar{w}$. When the dimension is large, this loss is quite significant. The culprit is obviously that GD is not forcing the initial weights to go to zero when there is no data from irrelevant and unused dimensions. This simple problem illustrates how the optimizer and an over-parameterized model might interact and lead to a solution that is prone to attacks.

An effective solution is to regularize the loss such that the weights of irrelevant dimensions to go to zero. Generic regularizers such as ridge and Lasso regression lead to a biased estimate of w_1^*, and thus, one is motivated to define a regularizer that is specially-designed for improving adversarial robustness. Bishop (1995) showed the close connection between training with random perturbation and Tikhonov Regularization. Inspired by this idea, we develop a regularizer that mimics the adversary itself. For this FGSM-like adversary, the population loss at the perturbed point is

$$\mathcal{L}_{\text{robustified}}(w) \triangleq \mathbb{E}\left[l(X + \Delta x; w)\right] = \mathcal{L}(w) + \varepsilon \mathbb{E}\left[r(X; w)\right] \left\|w_{2:d}\right\|_1 + \frac{\varepsilon^2}{2} \left\|w_{2:d}\right\|_1^2 . \tag{1}$$

Minimizing $\mathcal{L}_{\text{robustified}}(w)$ is equivalent to minimizing the model at the point $x' = x + \Delta x$. The regularizer $\varepsilon \mathbb{E}\left[r(X; w)\right] \left\|w_{2:d}\right\|_1 + \frac{\varepsilon^2}{2} \left\|w_{2:d}\right\|_1^2$ incorporates the effect of adversary in exact form.

Nonetheless, there are two limitations of this approach. The first is that it is designed for a particular choice of attack, an FGSM-like one. We would like a regularizer that is robust to a larger class of attacks. The second is that this regularizer is designed for a linear model and the squared error loss. How can we design a regularizer for more complicated models, such as DNNs? We address these questions by formulating the problem of adversarial robustness within the robust optimization framework (Section 3), and propose an approach to approximately solve it (Section 4).

3 Robust Optimization Formulation

Designing an adversarial robust estimator can be formulated as a robust optimization problem (Huang et al., 2015; Madry et al., 2018; Wong & Kolter, 2018; Shaham et al., 2018). To describe it, let us introduce our notations first. Consider an input space $\mathcal{X} \subset \mathbb{R}^d$, an output space $\mathcal{Y}$, and a parameter (or hypothesis) space $\mathcal{W}$, parameterizing a model $f : \mathcal{X} \times \mathcal{W} \to \mathcal{Y}$. In the supervised learning scenario, we are given a data distribution $\mathcal{D}$ over pairs of examples $\{(X_i, Y_i)\}_{i=1}^n$. Given the prediction of $f(x; w)$ and a target value y, the pointwise loss function of the model is denoted by $\ell(x, y; w) \triangleq \ell(f(x; w), y)$. Given the distribution of data, one can define the population loss as $\mathcal{L}(w) = \mathbb{E}\left[\ell(X, Y; w)\right]$. The goal of the standard supervised learning problem is to find a $w \in \mathcal{W}$ that minimizes the population loss. A generic approach to do this is through empirical risk minimization (ERM). Explicit or implicit regularization is often used to control the complexity of the hypothesis to avoid over- or under-fitting (Hastie et al., 2009).

As shown in the previous section, it is possible to find a parameter w that minimizes the loss through ERM, but leads to a model that is vulnerable to adversarial examples. To incorporate the robustness notion in the model, it requires defenders to reconsider the training objective. It is also important to formalize and constrain the power of the adversary, so we understand the strength of the attack to which the model is resistant. This can be specified by limiting that the adversary can only modify any input x to $x + \delta$ with $\delta \in \Delta \subset \mathcal{X}$. Commonly used constraints are ε-balls w.r.t. the ℓ_p-norms, though other constraint sets have been used too (Wong et al., 2019b). This goal can be formulated as a robust optimization problem where the objective is to minimize the adversarial population loss given some perturbation constraint Δ:

$$\min_{w} \mathbb{E}_{(X,Y)\sim\mathcal{D}}\big[\max_{\delta\in\Delta} \ell(X + \delta, Y; w)\big] \tag{2}$$

We have an interplay between two goals: 1) the inner-max term looks for the worst-case loss around the input, while 2) the outer-min term optimizes the hypothesis by minimizing such a loss.

Note that solving the inner-max problem is often computationally difficult, so one may approximate it with a surrogate loss obtained from a particular attack. Adversarial training and its variants (Szegedy et al., 2013; Goodfellow et al., 2014; Kurakin et al., 2016; Madry et al., 2018; Wong et al., 2019a) can be intuitively understood as an approximation of this min-max problem via different $\delta(x)$.

As shown in Section 2, one can design a regularizer that provides the **exact** value of the loss function at the attacked point for a particular choice of model, loss function, and adversary, cf. (1). Such a regularizer relieved us from using a separate inner optimization procedure, as is done in adversarial training. Motivated by that example and the robust optimization framework discussed here, we develop a regularizer that can be understood as an upper-bound on the **worst-case** value of the loss at an attacked point under a second-order approximation of the loss function.

4 Second-Order Adversarial Regularizer (SOAR)

The main idea of SOAR is to approximate the loss function using the second-order Taylor series expansion around an input x and then solve the inner maximization term of the robust optimization formulation (2) using the approximated form. We show this for both ℓ_2 and ℓ_∞ attacks; the same idea can be applied to other ℓ_p norms. We describe crucial steps of the derivation in this section, and defer details to Appendix B .

Assuming that the loss is twice-differentiable, we can approximate the loss function around input x by the second-order Taylor expansion

$$\ell(x + \delta, y; w) \approx \tilde{\ell}_{\text{2nd}}(x + \delta, y; w) \triangleq \ell(x, y; w) + \nabla_x \ell(x, y; w)^\top \delta + \frac{1}{2}\delta^\top \nabla_x^2 \ell(x, y; w)\delta. \tag{3}$$

For brevity, we drop w, y and use ∇ to denote ∇_x. Let us focus on the ℓ_p attacks, where the constraint set in (2) is $\Delta = \{\delta : \|\delta\|_p \leq \varepsilon\}$ for some $\varepsilon > 0$ and $p \geq 1$. We focus on the ℓ_∞ attack because of its popularity, but we also derive the formulation for the ℓ_2 attacks.

As a warm-up, let us solve the inner optimization problem by considering the first-order Taylor series expansion. We have

$$\ell_{\text{FOAR}}(x) \triangleq \max_{\|\delta\|_\infty \leq \varepsilon} \ell(x) + \nabla \ell(x)^\top \delta = \ell(x) + \varepsilon \|\nabla \ell(x)\|_1 . \tag{4}$$

The term $\varepsilon \|\nabla \ell(x)\|_1$ defines the First-Order Adversarial Regularizer (FOAR). This is similar to the regularizer introduced by Simon-Gabriel et al. (2019) with the choice of ℓ_∞ perturbation set. For a general ℓ_p-attach with $1 \leq p \leq \infty$, we have $\|\nabla \ell(x)\|_q$ with q satisfying $p^{-1} + q^{-1} = 1$. We shall empirically evaluate FOAR-based approach (for the ℓ_∞ attack), but our focus is going to be on solving the inner maximization problem based on the second-order Taylor expansion:

$$\max_{\|\delta\|_p \leq \varepsilon} \ell(x) + \nabla \ell(x)^\top \delta + \frac{1}{2} \delta^\top \nabla^2 \ell(x) \delta, \tag{5}$$

for $p = 2, \infty$. The second-order expansion in (3) can be rewritten as

$$\ell(x+\delta) \approx \ell(x) + \frac{1}{2} \begin{bmatrix} \delta \\ 1 \end{bmatrix}^\top \begin{bmatrix} \nabla^2 \ell(x) & \nabla \ell(x) \\ \nabla \ell(x)^\top & 1 \end{bmatrix} \begin{bmatrix} \delta \\ 1 \end{bmatrix} - \frac{1}{2} = \ell(x) + \frac{1}{2} \delta'^\top \mathbf{H} \delta' - \frac{1}{2}, \tag{6}$$

where $\delta' = [\delta; 1]$. This allows us to derive an upper bound on the expansion terms using the characteristics of a single Hessian term $\mathbf{H}$. Note that δ' is a $d+1$-dimensional vector and $\mathbf{H}$ is a $(d+1) \times (d+1)$ matrix. We need to find an upper bound on $\delta'^\top \mathbf{H} \delta'$ under the attack constraint.

For the ℓ_∞ attack, solving this maximizing problem is not as easy as in (4) since the Boolean quadratic programming problem in formulation (5) is NP-hard. But we can relax the constraint set and find an upper bound for the maximizer. Note that with $\delta \in \mathbb{R}^d$, an ℓ_∞-ball of size ε is enclosed by an ℓ_2-ball of size $\sqrt{d}\varepsilon$ with the same centre. Therefore, we can upper bound the inner maximization by

$$\max_{\|\delta\|_\infty \leq \varepsilon} \ell(x+\delta) \leq \max_{\|\delta\|_2 \leq \sqrt{d}\varepsilon} \ell(x+\delta), \tag{7}$$

which after substituting the second-order Taylor series expansion leads to an ℓ_2-constrained quadratic optimization problem

$$\ell(x) + \frac{1}{2} \max_{\|\delta\|_2 \leq \sqrt{d}\varepsilon} \delta'^\top \mathbf{H} \delta' - \frac{1}{2}, \tag{8}$$

with $\delta' = [\delta; 1]$ as before. The ℓ_2 version of SOAR does not require this extra step, and we have ε instead of $\sqrt{d}\varepsilon$ in (8). Before presenting a proposition that suggests a way to solve this maximization problem, we would like to remark on the upper bounding of the $\varepsilon - \ell_\infty$ ball with an $\sqrt{d}\varepsilon - \ell_2$ ball. If δ is a sparse vector, this upper bound can be loose for large d. But it appears that in practice, this upper bound is not loose after all, as the attack δ is often a dense vector. A more detailed discussion is included in Appendix B.2 .

Proposition 1. *Let $\ell : \mathbb{R}^d \to \mathbb{R}$ be a twice-differentiable function. For any $\varepsilon > 0$, we have*

$$\max_{\|\delta\|_\infty \leq \varepsilon} \tilde{\ell}_{2nd}(x+\delta) \leq \ell(x) + \frac{d\varepsilon^2 + 1}{2} \mathbb{E}\left[\|\mathbf{H}z\|_2\right] - \frac{1}{2}, \tag{9}$$

where $\boldsymbol{H}$ is defined in (6) and $z \sim \mathcal{N}(0, \mathbf{I}_{(d+1)\times(d+1)})$.

This result upper bounds the maximum of the second-order approximation $\tilde{\ell}_{2\text{nd}}$ over an ℓ_∞ ball with radius ε, and relates it to an expectation of a Hessian-vector product. Note that there is a simple correspondence between (1) and regularized loss in (9). The latter can be understood as an upper bound on the worst-case damage of an adversary under a second-order approximation of the loss. For the ℓ_2 attack, the same line of argument leads to $\varepsilon^2 + 1$ instead of $d\varepsilon^2 + 1$.

Let us take a closer look at $\mathbf{H}z$. By decomposing $z = [z_d, z_1]^\top$, we get

$$\mathbf{H}z = \begin{bmatrix} \nabla^2 \ell(x) z_d + z_1 \nabla \ell(x) \\ \nabla \ell(x)^\top z_d + z_1 \end{bmatrix}.$$

The term $\nabla^2 \ell(x) z_d$ can be computed using Finite Difference (FD) approximation. Note that $\mathbb{E}\left[\|z_d\|_2\right] = \sqrt{d}$ for our Normally distributed z. To ensure that the approximation direction has the same magnitude, we use the normalized $\tilde{z}_d = \frac{z_d}{\|z_d\|_2}$ instead, and use the approximation below

$$\nabla^2 \ell(x) z_d \approx \|z_d\|_2 \frac{\nabla \ell(x + h\tilde{z}_d) - \nabla \ell(x)}{h}. \tag{10}$$

To summarize, SOAR regularizer evaluated at x, with a direction z, and FD step size $h > 0$ is

$$R(x; z, h) = \frac{d\varepsilon^2 + 1}{2} \left\| \begin{bmatrix} \|z_d\|_2 \frac{\nabla \ell(x + h\tilde{z}_d) - \nabla \ell(x)}{h} + z_1 \nabla \ell(x) \\ \nabla \ell(x)^\top z_d + z_1 \end{bmatrix} \right\|_2. \tag{11}$$

The expectation in (9) can then be approximated by taking multiple samples of z drawn from $z \sim \mathcal{N}(0, \mathbf{I}_{(d+1)\times(d+1)})$. These samples would be concentrated around its expectation. One can show that $\mathbb{P}\left\{\|Hz\| - \mathbb{E}\left[\|Hz\|\right] > t\right\} \leq 2\exp(-\frac{ct^2}{\|H\|_2})$, where c is a constant and $\|H\|_2$ is the ℓ_2-induced norm (see Theorem 6.3.2 of Vershynin 2018). In practice, we observed that taking more than one sample of z do not provide significant improvement for increasing adversarial robustness.

Before we discuss the remaining details, recall that we fully robustify the model with an appropriate regularizer in Section 2. Note the maximizer of the loss based on formulation (2) is exactly the FGSM direction, and (1) shows the population loss with our FGSM-like choice of Δx. To further motivate a second-order approach, note that we can obtain the first two terms in (1) with a first-order regularizer such as FOAR; and we recover the exact form with a second-order formulation in (5).

Next, we study SOAR in the simple logistic regression setting, which shows potential failure of the regularizer and reveals why we might observe gradient masking. Based on that insight, we provide the remaining details of the method afterwards in Section 4.1.

4.1 Avoiding Gradient Masking

Consider a linear classifier $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ with $f(x; w) = \phi(\langle w, x \rangle)$, where $x, w \in \mathbb{R}^d$ are the input and the weights, and $\phi(\cdot)$ is the sigmoid function. Note that the output of f has the interpretation of being a Bernoulli distribution. For the cross-entropy loss function $\ell(x, y; w) = -[y \log f(x; w) + (1-y)\log(1 - f(x; w))]$, the gradient w.r.t. the input x is $\nabla \ell(x) = (f(x; w) - y)w$ and the Hessian w.r.t. the input x is $\nabla^2 \ell(x) = f(x; w)(1 - f(x; w))ww^\top$.

The second-order Taylor series expansion (3) with the gradient and Hessian evaluated at x is

$$\ell(x + \delta) \approx \ell(x) + r(x, y; w)w^\top \delta + \frac{1}{2} u(x; w)\delta^\top ww^\top \delta, \tag{12}$$

where $r = r(x, y; w) = f(x; w) - y$ is the residual term describing the difference between the predicted probability and the correct label, and $u = u(x; w) = f(x; w)(1 - f(x; w))$. Note that u can be interpreted as how confident the model is about its predication (correct or incorrect), and is close to 0 whenever the classifier is predicting a value close to 0 or 1. With this linear model, the maximization (8) becomes

$$\ell(x) + \max_{\|\delta\|_2 \leq \sqrt{d}\varepsilon} \left[rw^\top \delta + \frac{1}{2} u\delta^\top ww^\top \delta \right] = \ell(x) + \varepsilon\sqrt{d}\,|r(x, y; w)|\,\|w\|_2 + \frac{d\varepsilon^2}{2} u(x; w)\,\|w\|_2^2.$$

The regularization term is encouraging the norm of w to be small, weighted according to the residual $r(x, y; w)$ and the uncertainty $u(x; w)$. Whenever the classifier is correct with a high confidence, both r and u will be close to zero. As a result, the effect of the regularizer diminishes, i.e., the weights are no longer regularized. In such a case, the Taylor series expansion, computed using the gradient and Hessian evaluated at x, becomes an inaccurate approximation to the loss, and hence its maximizer is not a good solution to the inner maximization problem.

Note that this does not mean that one cannot use Taylor series expansion to approximate the loss. In fact, by the mean value theorem, there exists an $h^\star \in (0, 1)$ such that the second-order Taylor expansion is exact: $\ell(x + \delta) = \ell(x) + \nabla \ell(x)^\top \delta + \frac{1}{2}\delta^\top \nabla^2 \ell(x + h^\star \delta)\delta$. The issue is that if we compute the Hessian at x (instead of at $x + h^\star$), our approximation might not be very good whenever the loss function is close to being flat at x.

Algorithm 1: Computing the SOAR objective for a single training data

Input : A pair of training data (x, y), ℓ_∞ constraint of ε, Finite difference step-size h.

1 $x' \leftarrow x + \eta$, where $\eta \leftarrow (\eta_1, \eta_2, \ldots, \eta_d)^\top$ and $\eta_i \sim \mathcal{U}(-\frac{\varepsilon}{2}, \frac{\varepsilon}{2})$.

2 $x' \leftarrow \Pi_{B(x, \frac{\varepsilon}{2})} \left\{ x' + \frac{\varepsilon}{2} \operatorname{sign}\left(\nabla_x \ell(x')\right) \right\}$ where Π is the projection operator.

3 Sample $z \sim \mathcal{N}(0, \mathbf{I}_{(d+1)\times(d+1)})$.

4 Compute SOAR regularizer $R(x'; z, h)$ as (11).

5 Compute the pointwise objective: $\ell_{\text{SOAR}}(x, y) = \ell(x', y) + R(x'; z, h)$.

More importantly, a method relying on the gradient masking can be easily circumvented (Athalye et al., 2018). Our early experimental results had also indicated that gradient masking occurred with SOAR when the gradient and Hessian were evaluated at x. In particular, we observe that SOAR with zero-initialization leads to models with nearly 100% confidence on their predictions, leading to an ineffective regularizer. The result is reported in Table 3 in Appendix D.

This suggests a heuristic to improve the quality of SOAR. That is to evaluate the gradient and Hessian, through FD approximation (10) at a less confident point in the ℓ_∞ ball of x. We found that evaluating the gradient and Hessian at 1-step PGD adversary successfully circumvent the issue (Line 1-2 in Algorithm 1). We compare other initializations in Table 3 in Appendix D. To ensure the regularization is of the original ℓ_∞ ball of ε, we use $\frac{\varepsilon}{2}$ for PGD1 initialization, and then $\frac{\varepsilon}{2}$ in SOAR.

Based on this heuristic, the regularized pointwise objective for a data point (x, y) is

$$\ell_{\text{SOAR}}(x, y) = \ell(x', y) + R(x'; z, h), \tag{13}$$

where $z \sim \mathcal{N}(0, \mathbf{I}_{(d+1)\times(d+1)})$ and the point x' is initialized at PGD1 adversary. Algorithm 1 summarizes SOAR on a single training data. We include the full training procedure in Appendix C . Moreover, we include additional discussions and experiments on gradient masking in Appendix E.9.

4.2 Related Work

Several regularization-based alternatives to adversarial training have been proposed. Simon-Gabriel et al. (2019) studied regularization under the first-order Taylor approximation. The proposed regularizer for the ℓ_∞ perturbation set is the same as FOAR. Qin et al. (2019) propose local linearity regularization (LLR), where the local linearity measure is defined by the maximum error of the first-order Taylor approximation of the loss. LLR minimizes the local linearity mesaure, and minimizes the magnitude of the projection of gradient along the corresponding direction of the local linearity mesaure. It is motivated by the observation of flat loss surfaces during adversarial training.

CURE (Moosavi-Dezfooli et al., 2019) is the closest to our method. They empirically observed that adversarial training leads to a reduction in the magnitude of eigenvalues of the Hessian w.r.t. the input. Thus, they proposed directly minimizing the curvature of the loss function to mimic the effect of adversarial training. An important advantage of our proposed method is that SOAR is derived from a complete second-order Taylor approximation of the loss, while CURE exclusively focuses on the second-order term for the estimation of the curvature. Note the final optimization objective in SOAR, FOAR, LLR and CURE contains derivative w.r.t. the input of the DNN, and such a technique was first introduced to improve generalization by Drucker & Le Cun (1992) as double backpropagation.

Another related line of adversarial regularization methods do not involve approximation to the loss function nor robust optimization. TRADES (Zhang et al., 2019) introduces a regularization term that penalizes the difference between the output of the model on a training data and its corresponding adversarial example. MART (Wang et al., 2020) reformulated the training objective by explicitly differentiating between the mis-classified and correctly classified examples. Ding et al. (2019b) present another regularization approach that leverages adaptive margin maximization (MMA) on correctly classified example to robustify the model.

Table 1: Performance on CIFAR-10 against ℓ_∞ bounded white-box attacks ($\varepsilon = 8/255$).

	Method	Standard	FGSM	PGD20	PGD100	PGD200	PGD1000	PGD20-50
WideResNet	Standard	**95.79%**	44.77%	0.03%	0.01%	0.00%	0.00%	0.00%
	ADV	<u>87.14</u>%	55.38%	45.64%	45.05%	45.01%	44.95%	45.21%
	TRADES	84.92%	60.87%	**55.40%**	**55.10%**	**55.11%**	**55.06%**	**55.05%**
	MART	83.62%	**61.61%**	**56.29%**	**56.11%**	**56.10%**	**56.07%**	**55.98%**
	MMA	84.36%	**61.97%**	52.01%	51.26%	51.28%	51.02%	50.93%
ResNet	Standard	**92.54%**	21.59%	0.14%	0.09%	0.10%	0.08%	0.10%
	ADV	80.64%	50.96%	42.86%	42.27%	42.21%	42.17%	42.55%
	TRADES	75.61%	50.06%	45.38%	45.19%	45.18%	45.16%	45.24%
	MART	75.88%	52.55%	46.60%	46.29%	46.25%	46.21%	46.40%
	MMA	82.37%	47.08%	37.26%	36.71%	36.66%	36.64%	36.85%
	FOAR	65.84%	36.96%	32.28%	31.87%	31.89%	31.89%	32.08%
	SOAR	<u>87.95</u>%	**67.15%**	**56.06%**	**55.00%**	**54.94%**	**54.69%**	**54.20%**

5 Experiments

In this section, we verify the effectiveness of the proposed regularization method against ℓ_∞ PGD attacks on CIFAR10. Our experiments show that training with SOAR leads to significant improvements in adversarial robustness, which is achieved without significantly sacrificing standard accuracy. We focus on ℓ_∞ in this section and defer evaluations on ℓ_2 in Appendix E.4. Additionally, we provide a detailed discussion and evaluations on the SVHN dataset in Appendix E.5. Finally, we stress that provable robustness as in (Wong & Kolter, 2018) is not the focus of our work and the experiment result only provides an empirical evidence of the robustness.

We train ResNet-10 (He et al., 2016) on the CIFAR-10 dataset. The baseline methods consist of: (1) Standard: training with no adversarially perturbed data; (2) ADV: training with 10-step PGD adversarial examples; (3) TRADES; (4) MART and (5) MMA. Empirical studies in Madry et al. (2018) and Wang et al. (2020) reveal that their approaches benefit from increasing model capacity to achieve higher adversarial robustness, as such, we include WideResNet (Zagoruyko & Komodakis, 2016) for all baseline methods. We were not able to reproduce the results of two closely related works, CURE and LLR, which we discuss further in Appendix E.1. In Appendix E.11, we compare SOAR and FOAR with different initializations. FOAR achieves the best adversarial robustness using PGD1 initialization, so we only present this variation of FOAR in this section.

The optimization procedure is described in detail in Appendix E.2. Note that all ℓ_∞-PGD adversarial examples used for all training and evaluations are generated with $\varepsilon = 8/255$ and a step size of $2/255$ (pixel values are normalized to $[0, 1]$). PGD20-50 denotes 20-step PGD attacks with 50 restarts. Results on ResNet10 are obtained by averaging over 3 independently initialized and trained models, where the standard deviations are reported in Appendix E.8. We use the provided pretrained WideResNet model provided in the public repository of each method. Additionally, we encountered a number of challenges while implementing SOAR which we discuss in details in Appendix E.6.

5.1 Robustness against White-Box Attacks

Before making the comparison between SOAR and the baselines in Table 1, note that FOAR achieves 32.28% against PGD20 attacks. Despite its uncompetitive performance, this shows that approximating the robust optimization formulation based on Taylor series expansion is a reasonable approach. Furthermore, this justifies our extension to a second-order approximation, as the first-order alone is not sufficient. Next, we observe that training with SOAR significantly improves the adversarial robustness against all types of attacks, leading to higher robustness in all attacks on the ResNet model. SOAR remains competitive compared to baseline methods trained on high-capacity WideResNet architecture. We emphasize the importance that *improving adversarial robustness should not come at the expense of losing standard accuracy*. Note that the best overall defence method

Table 2: Performance on CIFAR-10 against ℓ_∞ bounded black-box attacks ($\varepsilon = 8/255$).

	Method	SimBA	PGD20-R	PGD20-W	PGD1000-R	PGD1000-W
WideResNet	ADV	49.20%	**84.69%**	**86.30%**	**84.69%**	**86.24%**
	TRADES	58.97%	82.18%	83.88%	82.24%	83.95%
	MART	59.60%	80.79%	82.62%	80.96%	82.76%
	MMA	58.30%	80.73%	82.74%	80.76%	82.74%
ResNet	ADV	47.27%	77.19%	79.48%	77.22%	79.55%
	TRADES	47.67%	72.28%	74.39%	72.24%	74.37%
	MART	48.57%	72.99%	74.91%	72.99%	75.04%
	MMA	43.53%	78.70%	80.39%	78.72%	81.35%
	FOAR	35.97%	63.56%	65.20%	63.60%	65.27%
	SOAR	**68.57%**	**79.25%**	**86.35%**	**79.49%**	**86.47%**

MART on WideResNet achieves only 0.23% higher accuracy against PGD20 than SOAR on ResNet, while we achieve 4.33% higher accuracy in standard accuracy. Previous works have usually attributed the drop in standard accuracy to a trade-off between accuracy and robustness (Madry et al., 2018; Tsipras et al., 2018). Compared to the baselines, SOAR substantially improves this trade-off while retaining competitive adversarial robustness.

5.2 Robustness against Black-Box Attacks

Many defences only reach an illusion of robustness through methods collectively known as gradient masking (Athalye et al., 2018). These methods often fail against attacks generated from an undefended independently trained model, known as transfer-based black-box attacks. Recent works (Tramèr et al., 2017; Ilyas et al., 2019) have proposed hypotheses for the success of transfer-based black-box attacks. In our evaluation, the transferred attacks are PGD20 and PGD1000 perturbations generated from two source models: ResNet and WideResNet, which are denoted by the suffix -R and -W respectively. The source models are trained separately from the defence models on the unperturbed training set. Additionally, Tramer et al. (2020) recommends score-based black-box attacks such as SimBA (Guo et al., 2019). They are more relevant in real-world applications where gradient information is not accessible, and are empirically shown to be more effective than transfer-based attacks. Because they are solely based on the confidence score of the model, score-based attacks are resistant to gradient-masking. SimBA works by first randomly sampling a vector from a predefined orthonormal basis then either adding or subtracting the random vector from the target image based on model's confidence score. All black-box attacks are ℓ_∞ constrained at $\varepsilon = 8/255$.

SOAR achieves the best robustness against all baseline methods trained on ResNet, as shown in Table 2. Compared with the baselines trained on WideResNet, SOAR remains the most robust model against transferred PGD20-W and PGD1000-W, approaching its standard accuracy on unperturbed data. Note that all defence methods are substantially more vulnerable to the score-based SimBA attack. SOAR regularized model is the most robust method against SimBA.

6 Conclusion

This work proposed SOAR, a regularizer that improves the robustness of DNN to adversarial examples. SOAR was obtained using the second-order Taylor series approximation of the loss function w.r.t. the input, and approximately solving the inner maximization of the robust optimization formulation. We showed that training with SOAR leads to adversarial robustness, with significant improvement in robustness under white-box and black-box ℓ_∞ and $\ell_2$attacks. More importantly, SOAR achieves robustness gain without a significant trade-off with standard accuracy. This is only one step in designing better regularizers to improve the adversarial robustness. One future direction is to understand the loss surface of DNN better in order to select a good point around which an accurate Taylor approximation can be made. This is important for designing regularizers that are not affected by gradient masking.

REFERENCES

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 274–283, 2018. 6, 8, 24

John E Beasley. Heuristic algorithms for the unconstrained binary quadratic programming problem. *London, England*, 1998. 15

Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*, volume 28. Princeton University Press, 2009. 1

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009. 23

Chris M Bishop. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7 (1):108–116, 1995. 2, 13

Qi-Zhi Cai, Chang Liu, and Dawn Song. Curriculum adversarial training. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 3740–3747, 2018. 21, 23

Gavin Weiguang Ding, Kry Yik Chau Lui, Xiaomeng Jin, Luyu Wang, and Ruitong Huang. On the sensitivity of adversarial robustness to input data distributions. In *International Conference on Learning Representations*, 2019a. 12

Gavin Weiguang Ding, Yash Sharma, Kry Yik Chau Lui, and Ruitong Huang. Mma training: Direct input space margin maximization through adversarial training. In *International Conference on Learning Representations*, 2019b. 6

Gavin Weiguang Ding, Luyu Wang, and Xiaomeng Jin. AdverTorch v0.1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623*, 2019c. 20

Harris Drucker and Yann Le Cun. Improving generalization performance using double backpropagation. *IEEE Transactions on Neural Networks*, 3(6):991–997, 1992. 6

Angus Galloway, Anna Golubeva, Thomas Tanay, Medhat Moussa, and Graham W Taylor. Batch normalization is a cause of adversarial vulnerability. *arXiv preprint arXiv:1905.02161*, 2019. 23

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 3, 12

Chuan Guo, Jacob Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Weinberger. Simple black-box adversarial attacks. In *International Conference on Machine Learning*, pp. 2484–2493, 2019. 2, 8, 20

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd edition)*. Springer, 2009. 3

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, 2016. 7

Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári. Learning with a strong adversary. *arXiv preprint arXiv:1511.03034*, 2015. 3

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, pp. 125–136, 2019. 8

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015. 23

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016. 3

John E Laird. *The Soar cognitive architecture*. MIT press, 2012. 1

Ricardo M Lima and Ignacio E Grossmann. On the solution of nonconvex cardinality boolean quadratic programming problems: a computational study. *Computational Optimization and Applications*, 66(1):1–37, 2017. 15

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. 1, 3, 7, 8, 23

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Jonathan Uesato, and Pascal Frossard. Robustness via curvature regularization, and vice versa. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9078–9086, 2019. 6, 19

Chongli Qin, James Martens, Sven Gowal, Dilip Krishnan, Krishnamurthy Dvijotham, Alhussein Fawzi, Soham De, Robert Stanforth, and Pushmeet Kohli. Adversarial robustness through local linearization. In *Advances in Neural Information Processing Systems*, pp. 13824–13833, 2019. 6, 19

Sara Sabour, Yanshuai Cao, Fartash Faghri, and David J Fleet. Adversarial manipulation of deep representations. *arXiv preprint arXiv:1511.05122*, 2015. 12

Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In *Advances in Neural Information Processing Systems*, pp. 5014–5026, 2018. 1, 12

Uri Shaham, Yutaro Yamada, and Sahand Negahban. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing*, 307:195–204, 2018. 1, 3

Carl-Johann Simon-Gabriel, Yann Ollivier, Leon Bottou, Bernhard Schölkopf, and David Lopez-Paz. First-order adversarial vulnerability of neural networks and input dimension. In *International Conference on Machine Learning*, pp. 5809–5817, 2019. 4, 6, 21

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 23

Aman Sinha, Hongseok Namkoong, and John Duchi. Certifying some distributional robustness with principled adversarial training. In *International Conference on Learning Representations*, 2018. 1

Chawin Sitawarin, Supriyo Chakraborty, and David Wagner. Improving adversarial robustness through progressive hardening. *arXiv preprint arXiv:2003.09347*, 2020. 23

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. 1, 3

Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*, 2017. 8

Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. *arXiv preprint arXiv:2002.08347*, 2020. 8

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*, 2018. 8

Roman Vershynin. *High-dimensional probability*. Cambridge University Press: An Introduction with Applications in Data Science, 2018. 5

Jianyu Wang and Haichao Zhang. Bilateral adversarial training: Towards fast training of more robust models against adversarial attacks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6629–6638, 2019. 21

Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*, 2020. 6, 7, 23

Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pp. 5286–5295, 2018. 1, 3, 7

Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*, 2019a. 3, 23

Eric Wong, Frank Schmidt, and Zico Kolter. Wasserstein adversarial examples via projected sinkhorn iterations. In *International Conference on Machine Learning*, pp. 6808–6817, 2019b. 3

Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Ruslan Salakhutdinov, and Kamalika Chaudhuri. Adversarial robustness through local lipschitzness. *arXiv preprint arXiv:2003.02460*, 2020. 19

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 7

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pp. 7472–7482, 2019. 6, 19

A Derivations of Section 2: Linear Regression with an Over-parametrized Model

We derive the results reported in Section 2 in more detail here. Recall that we consider a linear model $f_w(x) = \langle w\,,\, x \rangle$ with $x, w \in \mathbb{R}^d$. We suppose that $w^* = (1, 0, 0, \ldots, 0)^\top$ and the distribution of $x \sim p$ is such that it is confined on a 1-dimensional subspace $\{\, (x_1, 0, 0, \ldots, 0) \,:\, x_1 \in \mathbb{R} \,\}$. So the density of x is $p\left((x_1, \ldots, x_d)\right) = p_1(x_1)\delta(x_2)\delta(x_3)\ldots\delta(x_d)$, where $\delta(\cdot)$ is Dirac's delta function.

We initialize the weights at the first time step as $w(0) \sim N(0, \sigma^2 \mathbf{I}_{d\times d})$, and use GD to find the minimizer of the population loss. The partial derivatives of the population loss are

$$\frac{\partial \mathcal{L}(w)}{\partial w_j} = \begin{cases} \int (w_1 - w_1^*) p_1(x_1) x \mathrm{d}x = (w_1 - w_1^*)\mu_1, \\ \int (w_j - w_j^*)\delta(x_j) x \mathrm{d}x = (w_j - w_j^*)0 = 0,\ j \neq 1. \end{cases}$$

where $\mu_1 = \mathbb{E}\left[X_1\right]$. Notice that the gradient in dimension $j = 1$ is non-zero, unless $(w_1 - w_1^*)\mu_1 = 0$. Assuming that $\mu_1 \neq 0$, this implies that the gradient won't be zero unless $w_1 = w_1^*$. On the other hand, the gradients in dimensions $j = 2, \ldots, d$ are all zero, so GD does not change the value of $w_j(t)$ for $j = 2, \ldots, d$. Therefore, under the proper choice of learning rate β, we get that the asymptotic solution of GD solution is $\bar{w} \triangleq \lim_{r\to\infty} w(t) = (w_1^*, w_2(0), w_3(0), \ldots, w_d(0))^\top$. It is clear that $\mathcal{L}(\bar{w}) = 0$, i.e., the population loss is zero, as noted already as our first observation in that section.

Also note that we can easily attack this model by perturbing x by $\Delta x = (0, \Delta x_2, \Delta x_3, \ldots, \Delta x_d)^\top$. The pointwise loss at $x + \Delta x$ is

$$l(x + \Delta x; w) = \frac{1}{2}\left|(w_1 - w_1^*)x_1 + \langle w\,,\, \Delta x \rangle\right|^2 = \frac{1}{2}\left|r(x; w) + \langle w\,,\, \Delta x \rangle\right|^2.$$

With the choice of $\Delta x_i = \varepsilon\,\mathrm{sign}(w_i(0))$ (for $i = 2, \ldots, d$) and $\Delta x_1 = 0$, an FGSM-like attack (Goodfellow et al., 2014) at the learned weight $\bar{w}$ leads to the pointwise loss of

$$l(x + \Delta x; \bar{w}) = \frac{1}{2}\varepsilon^2 \left[\sum_{j=2}^{d} |w_j(0)|\right]^2 \approx \frac{1}{2}\varepsilon^2 \left\|w(0)\right\|_1^2.$$

We comment that our choice of Δx is not from the same distribution as the training data x. This choice aligns with the hypotheses in Ding et al. (2019a); Schmidt et al. (2018) that adversarial examples come from a shifted data distribution; however, techniques such as feature adversaries (Sabour et al., 2015) focus on designing perturbations to be close to input distributions. We stress that the goal here is to illustrate the loss under this particular attack.

In order to get a better sense of this loss, we compute its expected value w.r.t. the randomness of weight initialization. We have that (including the extra $|w_1(0)|$ term too)

$$\mathbb{E}_{W\sim N(0,\sigma^2\mathbf{I}_{d\times d})}\left[\|W\|_1^2\right] = \mathbb{E}\left[\sum_{i,j=1}^{d} |W_i||W_j|\right] = \sum_{i=1}^{d}\mathbb{E}\left[|W_i|^2\right] + \sum_{i,j=1, i\neq j}^{d}\mathbb{E}\left[|W_i|\right]\mathbb{E}\left[|W_j|\right],$$

where we used the independence of the r.v. W_i and W_j when $i \neq j$. The expectation $\mathbb{E}\left[|W_i|^2\right]$ is the variance σ^2 of W_i. The r.v. $|W_j|$ has a folded normal distribution, and its expectation $\mathbb{E}\left[|W_j|\right]$ is $\sqrt{\frac{2}{\pi}}\sigma$. Thus, we get that

$$\mathbb{E}_{W\sim N(0,\sigma^2\mathbf{I}_{d\times 1})}\left[\|W\|_1^2\right] = d\sigma^2 + d(d-1)\frac{2}{\pi}\sigma^2 \approx \frac{2}{\pi}d^2\sigma^2,$$

for $d \gg 1$. The expected population loss of the specified attack Δx at the asymptotic solution $\bar{w}$ is

$$\mathbb{E}_{X,W}\left[l(X + \Delta x); \bar{w})\right] \approx O(\varepsilon^2 d^2 \sigma^2).$$

The dependence of this loss on dimension d is significant, showing that the learned model is quite vulnerable to attacks. We note that the conclusions would not change much with initial distributions other than the Normal distribution.

An effective solution is to regularize the loss to encourage the weights of irrelevant dimensions going to zero. A generic regularizer is to use the ℓ_2-norm of the weights, i.e., formulate the problem as a ridge regression. In that case, the regularized population loss is

$$\mathcal{L}_{\text{ridge}}(w) = \frac{1}{2}\mathbb{E}\left[|\langle X\,,\,w\rangle - \langle X\,,\,w^*\rangle|^2\right] + \frac{\lambda}{2}\|w\|_2^2\,.$$

One can see that the solution of $\nabla_w \mathcal{L}_{\text{ridge}}(w) = 0$ is $\bar{w}_1(\lambda) = \frac{\mu_1}{\mu_1+\lambda} w_1^*$ and $\bar{w}_j(\lambda) = 0$ for $j \neq 1$.

$$\bar{w}_j(\lambda) = \begin{cases} \frac{\mu_1}{\mu_1+\lambda} w_1^* & j = 1 \\ 0 & j \neq 1. \end{cases}$$

The use of this generic regularizer seems reasonable in this example, as it enforces the weights for dimensions 2 to d to become zero. Its only drawback is that it leads to a biased estimate of w_1^*. The bias, however, can be made small with a small choice for λ. We can obtain a similar conclusion for the ℓ_1 regularizer (Lasso).

As such, one has to define a regularizer that is specially-designed for improving adversarial robustness. Bishop (1995) showed the strong connection between training with random perturbation and Tikhonov Regularization. Inspired by this idea, we develop a regularizer that mimics the adversary itself. Let us assume that a particular adversary attacks the model by adding $\Delta x = (0, \varepsilon\,\text{sign}(w_2(0)), \ldots, \varepsilon\,\text{sign}(w_d(0))^\top$. The population loss at the perturbed point is

$$\begin{aligned}\mathcal{L}_{\text{robustified}}(w) \triangleq \mathbb{E}\left[l(X+\Delta x; w)\right] &= \frac{1}{2}\mathbb{E}\left[\left|r(x;w) + \varepsilon\sum_{j=2}^{d}|w_j|\right|^2\right] \\ &= \mathcal{L}(w) + \varepsilon\mathbb{E}\left[r(X;w)\right]\|w_{2:d}\|_1 + \frac{\varepsilon^2}{2}\|w_{2:d}\|_1^2\,,\end{aligned}$$

where $\|w_{2:d}\|_1 = \sum_{j=2}^{d}|w_j|$.[1] This is the same objective as (1) reported in Section 2. Note that minimizing $\mathcal{L}_{\text{robustified}}(w)$ is equivalent to minimizing the model at the point $x' = x + \Delta x$. The regularizer $\varepsilon\mathbb{E}\left[r(X;w)\right]\|w_{2:d}\|_1 + \frac{\varepsilon^2}{2}\|w_{2:d}\|_1^2$ incorporates the effect of adversary in exact form. This motivated the possibility of designing a regularized tailored to prevent attacks.

A.1 Derivation of the population loss under its first and second order apprxoimation

First, we show that the FGSM direction is the maximizer of the loss when the perturbation is ℓ_∞ constrained. Based on the pointwise loss at $x + \Delta x$, we have

$$\max_{\|\Delta X\|_\infty \le \varepsilon} l(x+\Delta x; w) = \frac{1}{2}\left|r(x;w) + \max_{\|\Delta X\|_\infty \le \varepsilon}\langle w\,,\,\Delta x\rangle\right|^2.$$

We use the Cauchy-Schwarz inequality to obtain

$$\max_{\|\Delta X\|_\infty \le \varepsilon}\langle w\,,\,\Delta x\rangle \le \max_{\|\Delta X\|_\infty \le \varepsilon}|\langle w\,,\,\Delta x\rangle| \le \max_{\|\Delta X\|_\infty \le \varepsilon}\|w\|_1\|\Delta x\|_\infty = \varepsilon\|w\|_1\,,$$

which leads to

$$\operatorname*{argmax}_{\|\Delta X\|_\infty \le \varepsilon} l(x+\Delta x; w) = \varepsilon\,\text{sign}(w).$$

Next, we show that the first-order approximation of $\mathbb{E}\left[l(X+\Delta x; w)\right]$ obtains the first two terms in (1).

Note the gradient of the loss w.r.t. the input is

$$\nabla_x l(x;w) = (\langle w\,,\,\Delta x\rangle - \langle w^*\,,\,\Delta x\rangle)(w - w^*) = r(x;w)(w - w^*),$$

[1] A similar, but more complicated result, would hold if the adversary could also attack the first dimension.

and the Hessian w.r.t. the input is

$$\nabla_x^2 l(x; w) = (w - w^*)(w - w^*)^\top.$$

The first-order Taylor series approximation is

$$\begin{aligned}
\mathcal{L}_{\text{robustified}}(w) \approx \hat{\mathcal{L}}_{\text{1st}}(w) &\triangleq \mathbb{E}\left[l(X; w) + \nabla_x l(X; w)^\top \Delta x\right] \\
&= \mathcal{L}(w) + \mathbb{E}\left[r(X; w)(w - w^*)^\top \Delta x\right] \\
&= \mathcal{L}(w) + \mathbb{E}\left[r(X; w) w^\top \Delta x\right] \\
&= \mathcal{L}(w) + \varepsilon \mathbb{E}\left[r(X; w)\right] \left\| w_{2:d} \right\|_1 .
\end{aligned}$$

Note that ${w^*}^\top \Delta x = 0$ because of our particular choice of Δx and w^*. Here we obtain the first two terms in (1).

The second-order Taylor series approximation is

$$\begin{aligned}
\mathcal{L}_{\text{robustified}}(w) \approx \hat{\mathcal{L}}_{\text{2nd}}(w) &\triangleq \mathbb{E}\left[l(X; w) + \nabla_x l(X; w)^\top \Delta x + \frac{1}{2} \Delta x^\top \nabla_x^2 l(x; w) \Delta x\right] \\
&= \mathcal{L}(w) + \varepsilon \mathbb{E}\left[r(X; w)\right] \left\| w_{2:d} \right\|_1 + \frac{1}{2} \Delta x^\top (w - w^*)(w - w^*)^\top \Delta x \\
&= \mathcal{L}(w) + \varepsilon \mathbb{E}\left[r(X; w)\right] \left\| w_{2:d} \right\|_1 + \frac{\varepsilon^2}{2} \left\| w_{2:d} \right\|_1^2 ,
\end{aligned}$$

which recovers the exact form in (1).

This completes the motivation of using second-order Taylor series approximation with our warm-up toy example.

B Derivations of Section 4: Second-Order Adversarial Regularizer (SOAR)

B.1 Relaxation

Note the Boolean quadratic programming (BQP) problem in formulation (5) is NP-hard (Beasley, 1998; Lima & Grossmann, 2017). Even though there exist semi-definite programming (SDP) relaxations, such approaches require the exact Hessian w.r.t. the input, which is computationally expensive to obtain for high-dimensional inputs. And even if we could compute the exact Hessian, SDP itself is a computationally expensive approach, and not suitable to be within the inner loop of a DNN training. As such, we relax the ℓ_∞ constraint to an ℓ_2 constraint, which as we see, leads to a computationally efficient solution.

B.2 The tightness of the bound in Eq (7)

In practice, the upper bound may not be loose after all. To explain this, note that the claim that an $\varepsilon - \ell_\infty$ ball is enclosed within an $\sqrt{d}\varepsilon - \ell_2$ ball follows from

$$\|\delta\|_2 = (\sum_{i=1}^{d} |\delta_i|^2)^{1/2} \leq ([\max_i |\delta_i|^2 \sum_{i=1}^{d} 1)^{1/2} = \sqrt{d}\, \|\delta\|_\infty .$$

This bound is loose when δ is sparse. It is tight, however, when δ is dense with components that are mostly $\pm \|\delta\|_\infty$. When all components are of this form, the inequality becomes an equality. This seems to be the type of δ that an ℓ_∞-based attack often generates.

For example, we empirically observed that for the ℓ_∞ PGD20 attack ($\varepsilon = \frac{8}{255}$) on a standard trained model on CIFAR10 testset, the average ℓ_2-norm of the perturbations is 1.37, which should be compared to $\sqrt{d}\varepsilon = 1.74$.

B.3 Unified Objective

We could maximize each term inside (8) separately and upper bound the max by $\max_{\|\delta\|_2 \leq \sqrt{d}\varepsilon} \nabla \ell(x)^\top \delta + \max_{\|\delta\|_2 \leq \sqrt{d}\varepsilon} \frac{1}{2} \delta^\top \nabla^2 \ell(x) \delta = \sqrt{d}\varepsilon \|\nabla \ell(x)\|_2 + \frac{1}{2} d\varepsilon^2 \sigma_{\max}(\nabla^2 \ell(x))$, where $\sigma_{\max}(\nabla^2 \ell(x))$ is the largest singular value of the Hessian matrix, $\nabla^2 \ell(x)$. Even though the norm of the gradient and the singular value of the Hessian have an intuitive appeal, separately optimizing these terms might lead to a looser upper bound than necessary. The reason is that the maximizer of the first two terms are $\operatorname{argmax} \left|\nabla \ell(x)^\top \delta\right| = \frac{\nabla \ell(x)}{\|\nabla \ell(x)\|_2}$ and the direction corresponding to the largest singular value of $\nabla^2 \ell(x)$. In general, these two directions are not aligned.

B.4 Proof of Proposition 1

Proof. By the inclusion of the ℓ_∞-ball of radius ε within the ℓ_2-ball of radius $\sqrt{d}\varepsilon$ and the definition of $\mathbf{H}$ in (6), we have

$$\begin{aligned}
\max_{\|\delta\|_\infty \leq \varepsilon} \tilde{\ell}_{\text{2nd}}(x) &\leq \max_{\|\delta\|_2 \leq \sqrt{d}\varepsilon} \tilde{\ell}_{\text{2nd}}(x) \\
&= \max_{\|\delta\|_2 \leq \sqrt{d}\varepsilon} \ell(x) + \frac{1}{2} \begin{bmatrix} \delta \\ 1 \end{bmatrix}^\top \begin{bmatrix} \nabla^2 \ell(x) & \nabla \ell(x) \\ \nabla \ell(x)^\top & 1 \end{bmatrix} \begin{bmatrix} \delta \\ 1 \end{bmatrix} - \frac{1}{2} \\
&= \ell(x) + \frac{1}{2} \max_{\|\delta\|_2 \leq \sqrt{d}\varepsilon} \begin{bmatrix} \delta \\ 1 \end{bmatrix}^\top \mathbf{H} \begin{bmatrix} \delta \\ 1 \end{bmatrix} - \frac{1}{2} \\
&\leq \ell(x) + \frac{1}{2} \max_{\|\delta'\|_2 \leq \sqrt{d\varepsilon^2 + 1}} \delta'^\top \mathbf{H} \delta' - \frac{1}{2}.
\end{aligned}$$

It remains to upper bound $\max_{\|\delta'\|_2 \leq \varepsilon'} \delta'^\top \mathbf{H} \delta'$ with $\varepsilon' = \sqrt{d\varepsilon^2 + 1}$. We use the Cauchy-Schwarz inequality to obtain

$$\max_{\|\delta'\|_2 \leq \varepsilon'} \delta'^\top \mathbf{H} \delta' \leq \max_{\|\delta'\|_2 \leq \varepsilon'} \left| \delta'^\top \mathbf{H} \delta' \right| \leq \max_{\|\delta'\|_2 \leq \varepsilon'} \|\delta'\|_2 \|\mathbf{H}\delta'\|_2 = \varepsilon' \max_{\|\delta'\|_2 \leq \varepsilon'} \|\mathbf{H}\delta'\|_2 = \varepsilon'^2 \|\mathbf{H}\|_2 ,$$

where the last equality is obtained using properties of the ℓ_2-induced matrix norm (this is the spectral norm). Since computing $\|\mathbf{H}\|_2$ would again require the exact input Hessian, and we would like to avoid it, we further upper bound the spectral norm by the Frobenius norm as

$$\|\mathbf{H}\|_2 = \sigma_{\max}(\mathbf{H}) \leq \|\mathbf{H}\|_{\mathrm{F}} .$$

The Frobenius norm itself satisfies

$$\|\mathbf{H}\|_{\mathrm{F}} = \sqrt{\mathrm{Tr}(\mathbf{H}^\top \mathbf{H})} = \mathbb{E}\left[\|\mathbf{H}z\|_2\right], \tag{14}$$

where $z \sim \mathcal{N}(0, \mathbf{I}_{(d+1)\times(d+1)})$. Therefore, we can estimate $\|\mathbf{H}\|_{\mathrm{F}}$ by sampling random vectors z and compute the sample average of $\|\mathbf{H}z\|_2$. □

C SOAR ALGROITHM: A COMPLETE ILLUSTRATION

In Algorithm 1, we present the inner-loop operation of SOAR using a single data point. Here we summarize the full training procedure with SOAR in Algorithm 2. Note that it is presented as if the optimizer is SGD, but we may use other optimizers as well.

Algorithm 2: Improving adversarial robustness via SOAR

Input : Training dataset. Learning rate β, training batch size b, number of iterations N, ℓ_∞ constraint of ε, Finite difference step-size h.

Initialize network with pre-trained weight w;
for $i \in \{0, 1, \ldots, N\}$ **do**
  Get mini-batch $B = \{(x_1, y_1), \cdots, (x_b, y_b)\}$ from the training set.
  for $j = 1, \ldots, m$ *(in parallel)* **do**
    $x'_j \leftarrow x_j + \eta$, where $\eta \leftarrow (\eta_1, \eta_2, \ldots, \eta_d)^\top$ and $\eta_i \sim \mathcal{U}(-\frac{\varepsilon}{2}, \frac{\varepsilon}{2})$.
    $x'_j \leftarrow \Pi_{B(x_j, \frac{\varepsilon}{2})} \left\{ x'_j + \frac{\varepsilon}{2} \operatorname{sign}\left(\nabla_{x'_j} \ell(x'_j)\right) \right\}$ where Π is the projection operator.
    Sample $z \sim \mathcal{N}(0, \mathbf{I}_{(d+1)\times(d+1)})$.
    Compute the SOAR regularizer $R(x'_j; z, h)$ as (11).
    Compute the pointwise objective: $\ell_{\text{SOAR}}(x_j, y_j) = \ell(x'_j, y_j) + R(x'_j; z, h)$.
  end
  $w_{i+1} \leftarrow w_i - \beta \times \frac{1}{b} \sum_{j=1}^{b} \nabla_{w_i} \ell_{\text{SOAR}}$.
end

D POTENTIAL CAUSES OF GRADIENT MASKING

Table 3: Average value of the highest probability output for all test set data, that is, $\frac{1}{N}\Sigma_{n=1}^{N} \max_{i \in 1,2,\ldots,c} P(x_n)_i$, where $P(x_n)_i$ represent the probability of class i given data x_n.

Method	Standard	Random	PGD1
Standard	$98.11\% \pm 0.07$	$97.81\% \pm 0.06$	$96.83\% \pm 0.11$
ADV	$70.33\% \pm 0.45$	$70.04\% \pm 0.45$	$65.46\% \pm 0.40$
SOAR			
- zero init	$99.99\% \pm 0.01$	$\mathbf{99.97}\% \pm 0.01$	$99.99\% \pm 0.01$
- random init	$99.98\% \pm 0.01$	$\mathbf{99.98}\% \pm 0.00$	$100.0\% \pm 0.00$
- PGD1 init	$97.71\% \pm 0.10$	$97.63\% \pm 0.08$	$97.94\% \pm 0.09$

We summarize the average value of the highest probability output for test set data initialized with zero, random and PGD1 perturbations in Table 3. We notice that training with SOAR using zero or random initialization leads to models with nearly 100% confidence on their predictions. This is aligned with the analysis of SOAR for a linear classifier (Section 4.1), which shows that the regularizer becomes ineffective as the model outputs high confidence predictions. Indeed, results in Table 5 show that those models are vulnerable under black-box attacks.

Results in Table 3 suggest that highly confident predictions *could be* an indication for gradient masking. We demonstrate this using the gradient-based PGD attack. Recall that we generate PGD attacks by first initializing the clean data x_n with a randomly chosen η within the ℓ_∞ ball of size ε, followed by gradient ascent at $x_n + \eta$. Suppose that the model makes predictions with 100% confidence on any given input. This leads to a piece-wise loss surface that is either zero (correct predictions) or infinity (incorrect predictions). The gradient of this loss function is either zero or undefined, and thus making gradient ascent ineffective. Therefore, white-box gradient-based attacks are unable to find adversarial examples.

E SUPPLEMENTARY EXPERIMENTS

E.1 DISCUSSION ON THE REPRODUCIBILITY OF CURE AND LLR

We were not able to reproduce results of two closely related works, CURE (Moosavi-Dezfooli et al., 2019) and LLR (Qin et al., 2019). For CURE, we found the open-source implementation[2], but were not able to reproduce their reported results using their implmentation. We were not able to reproduce the results of CURE with our own implementation either. For LLR, Yang et al. (2020) were not able to reproduce the results, they also provided an open-source implementation[3]. Regardless, we compare SOAR to the **reported result** by CURE and LLR in Table 4:

Table 4: Comparison of SOAR, CURE and LLR on CIFAR-10 against ℓ_∞ bounded adversarial perturbations ($\varepsilon = 8/255$).

Method	Standard Accuracy	PGD20	Architecture
SOAR	87.95%	56.06%	ResNet-10
CURE	81.20%	36.30%	ResNet-18
CURE	83.10%	41.40%	WideResNet
LLR	86.83%	54.24%	WideResNet

E.2 TRAINING AND EVALUATION SETUP

CIFAR-10: Training data is augmented with random crops and horizontal flips.

ResNet: We used an open-source ResNet-10 implementation[4]. More specifically, we initialize the model with `ResNet(BasicBlock, [1,1,1,1])`. Note that we remove the BatchNorm layers in the ResNet-10 architecture, and we discuss this further in Appendix E.6 .

WideResNet: We used the implementation[5] of WideResNet-34-10 model found in public repository maintained by the authors of TRADES (Zhang et al., 2019).

Standard training on ResNet and WideResNet: Both are trained for a total of 200 epochs, with an initial learning rate of 0.1. The learning rate decays by an order of magnitude at epoch 100 and 150. We used a minibatch size of 128 for testing and training. We used SGD optimizer with momentum of 0.9 and a weight decay of 2e-4.

Adversarial training with PGD10 examples on ResNet: The optimization setting is the same as the one used for standard training. Additionally, to ensure that the final model has the highest adversarial robustness, we save the model at the end of every epoch, and the final evaluation is based on the one with the highest PGD20 accuracy.

SOAR on ResNet: SOAR refers to continuing the training of the Standard model on ResNet. It is trained for a total of 200 epochs with an initial learning rate of 0.004 and decay by an order of magnitude at epoch 100. We used SGD optimizer with momentum of 0.9 and a weight decay of 2e-4. We use a FD step-size $h = 0.01$ for the regularizer. Additionally, we apply a clipping of 10 on the regularizer, and we discuss this clipping operation in Appendix E.6 .

MART and TRADES on ResNet: We used the same optimization setup as the ones in their respective public repository[6]. We briefly summarize it here. The model is trained for a total of 120 epochs, with an initial learning rate of 0.1. The learning rate decays by an order of magnitude at epoch 75, 90, 100. We used SGD optimizer with momentum of 0.9 and a weight decay of 2e-4. We performed a hyperparameter sweep on the strength of the regularization term β and selected one that resulted in the best performance against PGD20 attacks. A complete result is reported in Appendix E.10 .

[2] https://github.com/F-Salehi/CURE_robustness
[3] https://github.com/yangarbiter/robust-local-lipschitz
[4] https://github.com/kuangliu/pytorch-cifar
[5] https://github.com/yaodongyu/TRADES
[6] https://github.com/YisenWang/MART

MMA on ResNet: We used the same optimization setup as the one in its public repository[7]. We briefly summarize it here. The model is trained for a total of 50000 iterations, with an initial learning rate of 0.3. The learning rate changes to 0.09 at the 20000 iteration, 0.03 at the 30000 iteration and lastly 0.009 at the 40000 iteration. We used SGD optimizer with momentum of 0.9 and a weight decay of 2e-4. We performed a hyperparameter sweep on the margin term and selected the one that resulted in the best performance against PGD20 attacks. A complete result is reported in Appendix E.10 .

ADV, TRADES, MART and MMA on WideResNet: We use the pretrained checkpoint provided in their respective repositories. Note that we use the pretrained checkpoint for PGD10 adversarially trained WideResNet in Madry's CIFAR10 Challenge[8].

Evaluations: For FGSM and PGD attacks, we use the implementation in AdverTorch (Ding et al., 2019c). For SimBA (Guo et al., 2019), we use the authors' open-source implementation[9].

E.3 Adversarial Robustness of the Model Trained using SOAR with Different Initializations

Table 5: Performance of SOAR with different initializations on CIFAR-10 against white-box and transfer-based black-box ℓ_∞ bounded adversarial perturbations ($\varepsilon = 8/255$).

Method	Standard accuracy	White-box PGD20	Black-box PGD20
SOAR			
- zero init	91.73%	89.24%	2.86%
- rand init	91.70%	90.82%	9.16%
- PGD1 init	87.95%	**56.06%**	**79.25%**

We report the adversarial robustness of the model trained using SOAR with different initialization techniques in Table 5. The second column shows the accuracy against white-box PGD20 adversaries. The third column shows the accuracy against black-box PGD20 adversaries transferred from an independently initialized and standard-trained ResNet-10 model.
Note that despite the high adversarial accuracy against white-box PGD attacks, models trained using SOAR with zero and random initialization perform poorly against transferred attacks. This suggests the presence of gradient masking when using SOAR with zero and random initializations. Evidently, SOAR with PGD1 initialization alleviates the gradient masking problem.

[7] https://github.com/BorealisAI/mma_training
[8] https://github.com/MadryLab/cifar10_challenge
[9] https://github.com/cg563/simple-blackbox-attack

E.4 ROBUSTNESS UNDER ℓ_2 ATTACKS ON CIFAR-10

We evaluate SOAR and two of the baseline methods, ADV and TRADES, against ℓ_2 white-box and black-box attacks on CIFAR-10 in Table 6. No ℓ_2 results were reported by MART and we are not able to reproduce the ℓ_2 results using the implementation by MMA, thus those two methods are not included in our evaluation.

In Section 4, we show that the ℓ_∞ formulation of SOAR with $\|\delta\|_\infty = \varepsilon$ is **equivalent** to the ℓ_2 formulation of SOAR with $\|\delta\|_2 = \varepsilon\sqrt{d}$. In other words, models trained with SOAR to be robust against ℓ_∞ attacks with $\varepsilon = \frac{8}{255}$ should also obtain improved robustness against ℓ_2 attacks with $\varepsilon = \frac{8}{255}\sqrt{32 * 32 * 3} = 1.74$. In our evaluation, all ℓ_2 adversaries used during ADV and TRADES are generated with 10-step PGD ($\varepsilon = 1.74$) and a step size of 0.44. Note that the goal here is to show the improved robustness of SOAR against ℓ_2 attacks other than being SOTA, thus the optimization procedures are the same as the ones used in the ℓ_∞ evaluation.

We observe that training with SOAR improves the robustness of the model against ℓ_2 attacks. Instead of a fixed ℓ_2 norm, we demonstrate the improved robustness using an increasing range of ε. For all attacks, we use 100 iterations of PGD and a step size of $\frac{2.5\varepsilon}{100}$. In Table 6, we find that training with SOAR leads to a significant increase in robustness against white-box and black-box ℓ_2 adversaries. As ε increases, SOAR model remain robust against white-box ℓ_2 attacks ($\varepsilon = 1$), while other methods falls off. The last column of Table 6 shows the robustness against transferred ℓ_2 attacks ($\varepsilon = 1.74$). The source model is a ResNet10 network trained separately from the defence models on the unperturbed training set. We observe that SOAR achieves the second highest robustness compared to baseline methods against transferred ℓ_2 attacks. This result empirically verifies our previous claim that ℓ_2 and ℓ_∞ formulation of SOAR only differs by a factor of $\sqrt{d}$. Moreover, it aligns with findings by Simon-Gabriel et al. (2019), that empirically showed adversarial robustness through regularization gains robustness against more than one norm-ball attack at the same time.

Table 6: ℓ_2 robustness of the adversarially trained model (under ℓ_2 formulations) at different epsilon values. 100-step PGD is used for all attacks. Accuracy (%) against ℓ_2-PGD attacks.

	Method	$\varepsilon = \frac{60}{255}$	$\varepsilon = \frac{120}{255}$	$\varepsilon = \frac{255}{255}$	$\varepsilon = 1.739$	Transfer
ResNet	ADV	68.13%	62.03%	47.53%	28.09%	**70.86%**
	TRADES	68.59%	62.39%	45.42%	25.67%	69.02%
	SOAR	**75.39%**	**66.81%**	**60.90%**	**56.89%**	<u>69.52</u>%

E.5 ADDITIONAL EVALUATION ON SVHN DATASET

We use the same ResNet-10 architecture as the one for CIFAR-10 evaluation. Training data is augmented with random crops and horizontal flips. For Standard training, we use the same optimization procedure as the one used for CIFAR-10. For SOAR and TRADES, we use the exact same hyperparameter for the regularizer. For SOAR, we use early-stopping at epoch 130 to prevent catastrophic over-fitting. Besides, the optimization schedule is identical for SOAR and TRADES as the ones used for CIFAR-10.

We emphasize again that the goal of evaluting using SVHN is to demonstrate the improved robustness with SOAR on a different dataset, thus we did not perform an additional hyper-parameter sweep. The optimization procedures are the same as the ones used in the CIFAR-10 evaluation.

For PGD10 adversarial training, we observe that ResNet-10 is not able to learn anything meaningful. Specifically, when trained with PGD10 examples, ResNet-10 does not perform better than a randomly-initialized network in both standard and adversarial accuracy. Cai et al. (2018) made a similar observation on ResNet-50, where training accuracy is not improving over a long period of adversarial training with PGD10. They further investigated models with different capacities and found that even ResNet-50 might not be sufficiently deep for PGD10 adversarial training on SVHN. Wang & Zhang (2019) reported PGD10 adversarial training result on SVHN with WideResNet, which we include in Table 7.

Table 7: Performance on SVHN against ℓ_∞ bounded white-box attacks ($\varepsilon = 8/255$).

Method	Standard	FGSM	PGD20	PGD100	PGD200	PGD1000	PGD20-50
Standard	**94.93**%	34.41%	2.71%	2.27%	2.20%	2.09%	2.15%
ADV	91.8%	61.0%	43.2%	42.1%	—	—	—
TRADES	84.48%	58.01%	48.90%	48.15%	48.11%	48.05%	48.01%
SOAR	91.23%	**72.70**%	**58.99**%	**56.80**%	**56.56**%	**56.38**%	**56.70**%

Table 8: Performance on SVHN against ℓ_∞ bounded black-box attacks ($\varepsilon = 8/255$).

Method	SimBA	PGD20	PGD1000
TRADES	49.13%	75.35%	75.27%
SOAR	**64.20**%	**75.42**%	**75.81**%

For MART, we were not able to translate their CIFAR-10 results on SVHN. We performed the same hyperparameter sweep as the one in Table 14, as well as different optimization settings, but none resulted in a meaningful model. It is likely that the potential cause is the small capacity of ResNet-10. For MMA, the implementation included in its public repository is very specific to the CIFAR-10 dataset, so we did not include it in the comparison.

Overall, we observe a similar performance on SVHN vs. on CIFAR-10. Compared to the result in Table 1, we observe a slight increase in standard accuracy and robust accuracy for both SOAR and TRADES. In particular, the standard accuracy increases by 8.87% and 3.28%, and the PGD20 accuracy increases by 3.52% and 2.93% for TRADES and SOAR respectively. More notably, we observe on SVHN that SOAR regularized model gains robustness without significantly sacrificing its standard accuracy.

Table 8 compares the performance of SOAR to TRADES on SimBa and on transferred ℓ_∞ attacks. The evaluation setting for transferred attacks is identical to the one used for CIFAR-10, where we use an undefended independently trained ResNet-10 as the source model. Despite a smaller gap on the accuracy against transferred attacks, we see that SOAR regularized model yields a significant higher accuracy against the stronger SimBA attacks.

Note that we did not perform any extensive hyperparameter sweep on SVHN, and we simply took what worked on CIFAR-10. We stress that the goal is to demonstrate the effectiveness of SOAR, and its performance relative to other baseline methods.

Next, we evaluate SOAR and TRADES under ℓ_2 bounded white-box and black-box attacks. All ℓ_2 PGD adversaries are generated using the same method as the one in the evaluation for CIFAR-10. Also, we do not include ADV due to the same result discussed above. Our results show that training with SOAR significantly improves the robustness against ℓ_2 PGD white-box attacks compared to TRADES. For transferred attacks, TRADES and SOAR performs similarly.

Table 9: Performance on SVHN against ℓ_2 bounded white-box and black-box attacks. 100-step PGD is used for all attacks.

Method	Standard	$\varepsilon = \frac{60}{255}$	$\varepsilon = \frac{120}{255}$	$\varepsilon = \frac{255}{255}$	$\varepsilon = 1.739$	Transfer
Standard	**94.93**%	53.46%	30.04%	18.20%	14.60%	15.52%
TRADES	91.94%	55.17%	47.47%	32.34%	17.52%	**50.22**%
SOAR	91.23%	**82.35**%	**69.65**%	**56.70**%	**48.38**%	49.13%

E.6 CHALLENGES

Batch Normalization: We observe that networks with BatchNorm layers do not benefit from SOAR in adversarial robustness. Specifically, we performed an extensive hyper-parameter search for SOAR on networks with BatchNorm layers, and we were not able to achieve meaningful improvement in adversarial robustness. A related work by Galloway et al. (2019) focuses on the connection between BatchNorm and adversarial robustness. In particular, their results show that on VGG-based architecture (Simonyan & Zisserman, 2014), there is a significant gap in adversarial robustness between networks with and without BatchNorm layers under standard training. Needless to say, the interaction between SOAR and BatchNorm requires further investigations, and we consider this as an important future direction. As such, we use a small-capacity ResNet (ResNet-10) in our experiment, and modified it by removing its BatchNorm layers. Note that BatchNorm layers makes the training process less sensitive to hyperparameters (Ioffe & Szegedy, 2015), and removing them makes it difficult to train a very deep network such as WideResNet. As such, we did not perform SOAR on WideResNet.

Starting from pretrained model: We notice that it is difficult to train with SOAR on a newly-initialized model. Note that it is a common technique to perform fine-tuning on a pretrained model for a specific task. In CURE, regularization is performed after a model is first trained with a cross-entropy loss to reach a high accuracy on clean data. They call the process *adversarial fine-tuning*. Cai et al. (2018); Sitawarin et al. (2020) study the connection between curriculum learning (Bengio et al., 2009) and training using adversarial examples with increasing difficulties. Our idea is similar. The model is first optimized for an easier task (standard training), and then regularized for a related, but more difficult task (improving adversarial robustness). Since the model has been trained to minimize its standard loss, the loss gradient can be very small compared to the regularizer gradient, and thus we apply a clipping of 10 on the regularizer.

Catastrophic Over-fitting: We observe that when the model achieves a high adversarial accuracy and continues training for a long period of time, both the standard and adversarial accuracy drop significantly. A similar phenomenon was observed in (Cai et al., 2018; Wong et al., 2019a), which they refer to as *catastrophic forgetting* and *catastrophic over-fitting* respectively. Wong et al. (2019a) use early-stopping as a simple solution. We observe that with a large learning rate, the model reaches a high adversarial accuracy faster and catastrophic over-fitting happens sooner. As such, our solution is to fix the number of epochs to 200 and then carefully sweep over various learning rates to make sure that catastrophic over-fitting do not happen.

Discussion on Computation Cost: We emphasize that our primary goal is to propose regularization as an alternative approach to improving adversarial robustness. We discussed techniques towards an efficient implementation, however, there is still potential for a faster implementation. In our current implementation, a single epoch with WideResNet takes: 19 mins on PGD10 adversarial training, 26.5 mins on SOAR, 29 mins on MART, and 39.6 mins on TRADES. We observe that despite being a faster method than MART and TRADES, SOAR is still quite slow compared to PGD10 adversarial training. Finally, compared to no initialization or random initialization, the overhead for the proposed PGD1 initialization requires one additional backward/forward pass.

E.7 POTENTIAL ROBUSTNESS GAIN WITH INCREASING CAPACITIES

Empirical studies in Madry et al. (2018) and Wang et al. (2020) reveal that their approaches benefit from increasing model capacity to achieve higher adversarial robustness. We have a similar observation with SOAR.

Table 10: Performance on CIFAR-10 against ℓ_∞ bounded white-box attacks ($\varepsilon = 8/255$).

Model	Standard	PGD20	PGD100	PGD200
CNN6	81.73%	32.83%	31.20%	31.15%
CNN8	83.65%	47.30%	46.07%	45.83%
ResNet-10	87.95%	56.06%	55.00%	54.94%

Table 10 compares the performance of SOAR against ℓ_∞ bounded white-box attacks on networks with different capacities. CNN6(CNN8) refers to a simple 6-layer(8-layer) convolutional network, and ResNet-10 is the network we use in Section 5. Evidently, as network capacity increases, we observe improvements in both standard accuracy and adversarial accuracy. As such, we expect a similar gain in performance with larger capacity networks such as WideResNet.

E.8 Experiment results on ResNet10 in Table 1 and Table 2 with standard deviaitions

All results on ResNet10 are obtained by averaging over 3 independently initialized and trained models. Here, we report the standard deviation of the results in Table 1 and Table 2. Notice we omit results on PGD100 and PGD200 due to space constraint.

Table 11: Performance on CIFAR-10 against ℓ_∞ bounded white-box attacks ($\varepsilon = 8/255$). (Table 1 with standard deviations)

	Method	Standard	FGSM	PGD20	PGD1000	PGD20-50
ResNet	Standard	92.54%(0.07)	21.59%(0.29)	0.14%(0.03)	0.08%(0.02)	0.10%(0.02)
	ADV	80.64%(0.40)	50.96%(0.27)	42.86%(0.18)	42.17%(0.15)	42.55%(0.15)
	TRADES	75.61%(0.19)	50.06%(0.09)	45.38%(0.15)	45.16%(0.17)	45.24%(0.16)
	MART	75.88%(1.12)	52.55%(0.28)	46.60%(0.43)	46.21%(0.47)	46.40%(0.48)
	MMA	82.37%(0.15)	47.08%(0.37)	37.26%(0.38)	36.64%(0.41)	36.85%(0.44)
	FOAR	65.84%(0.24)	36.96%(0.17)	32.28%(0.11)	31.89%(0.10)	32.08%(0.07)
	SOAR	87.95%(0.08)	67.15%(0.32)	56.06%(0.15)	54.69%(0.17)	54.20%(0.16)

Table 12: Performance on CIFAR-10 against ℓ_∞ bounded black-box attacks ($\varepsilon = 8/255$). (Table 2 with standard deviations)

	Method	SimBA	PGD20-R	PGD20-W	PGD1000-R	PGD1000-W
ResNet	ADV	47.27%(1.01)	77.19%(0.30)	79.48%(0.38)	77.22%(0.34)	79.55%(0.37)
	TRADES	47.67%(0.52)	72.28%(0.19)	74.39%(0.28)	72.24%(0.14)	74.37%(0.21)
	MART	48.57%(0.99)	72.99%(0.90)	74.91%(1.15)	72.99%(0.87)	75.04%(1.12)
	MMA	43.53%(1.25)	78.70%(0.09)	80.39%(1.20)	78.72%(0.09)	81.35%(0.17)
	FOAR	35.97%(0.26)	63.56%(0.30)	65.20%(0.21)	63.60%(0.33)	65.27%(0.32)
	SOAR	68.57%(0.95)	79.25%(0.45)	86.35%(0.04)	79.49%(0.29)	86.47%(0.10)

E.9 Additional Experiments on Gradient Masking

To verify that SOAR improves robustness of the model without gradient masking, we include the following experiments to empirically support our claim.

First, from the result in Appendix E.3, we conclude that SOAR with zero initilaization results in gradient masking. This is shown by the high accuracy (89.24%, close to standard accuracy) under white-box PGD attacks and low accuracy (2.86%) under black-box transferred attacks. Next, prior work has verified that adversarial training with PGD20 adversaries (ADV) results model without gradient masking Athalye et al. (2018). Therefore, let us use models trained using ADV and SOAR(zero-init) as examples of models with/without gradient masking respectively.

In the ℓ_∞ attack setting, PGD uses the sign of the gradient $\text{sign}(\nabla_x \ell(x))$ to generate perturbations. As such, one way to verify the strength of gradient is to measure the average number of none-zero elements in the gradient. A model with gradient masking is expected to have much less non-zero elements than one without. In our experiment, the average non-zero element in gradient is 3072 for ADV trained (no GM), 3069 for SOAR (PGD1-init) and 1043 for SOAR (zero-init, has GM). We observe that SOAR with PGD1-init has a similar number of non-zero gradient elements compared

to ADV, meaning PGD adversary can use sign of those non-zero gradient elements to generate meaningful perturbations.

In Section 5, the 20-iteration ℓ_∞ PGD adversaries are generated with a step-size of $\frac{2}{255}$ and $\varepsilon = \frac{8}{255}$. Suppose we use $\varepsilon = 1$ instead of $\varepsilon = \frac{8}{255}$ and other parameters remain the same, that is, we allow the maximum ℓ_∞ perturbation to reach the input range ($[0, 1]$) and generate PGD20 attacks. We observe such attacks result in near black-and-white images on SOAR with PGD-1 init; it has a 0% accuracy against such PGD20 attacks, similar to the 3.3% on ADV trained model. On the other hand, the robust accuracy for SOAR (zero-init) is 9.7%.

E.10 Hyperparameter sweep for TRADES, MART and MMA on ResNet

The following results show the hyperparameter sweep on TRADES, MART and MMA respectively. We include the one with the highest PGD20 accuracy in Section 5.

Table 13: Hyperparameter sweep of TRADES on ResNet: evaluation based performance on CIFAR-10 against ℓ_∞ bounded adversarial perturbations ($\varepsilon = 8/255$).

β	Standard	FGSM	PGD20	PGD100	PGD200	PGD20-50
15	72.80%	48.85%	44.80%	44.63%	44.63%	44.67%
13	73.46%	49.18%	45.09%	44.97%	44.96%	44.98%
11	74.46%	49.39%	44.86%	44.74%	44.71%	44.75%
9	75.61%	**50.06%**	**45.38%**	**45.19%**	**45.18%**	**45.24%**
7	76.74%	50.05%	45.04%	44.80%	44.78%	44.87%
5	78.39%	50.22%	44.40%	44.11%	44.14%	44.26%
3	80.20%	49.84%	42.53%	42.12%	42.09%	42.23%
1	83.69%	45.61%	35.27%	34.62%	34.52%	34.87%
0.5	84.00%	46.16%	34.40%	33.48%	33.38%	33.75%
0.1	84.64%	41.12%	16.52%	14.80%	14.53%	14.83%

Table 14: Hyperparameter sweep of MART on ResNet: evaluation based performance on CIFAR-10 against ℓ_∞ bounded adversarial perturbations ($\varepsilon = 8/255$).

β	Standard	FGSM	PGD20	PGD100	PGD200	PGD20-50
15	72.78%	52.13%	46.41%	46.05%	45.98%	46.14%
13	73.03%	51.43%	46.29%	46.05%	45.99%	46.15%
11	74.25%	51.57%	46.17%	45.84%	45.83%	45.95%
9	75.88%	**52.55%**	**46.60%**	**46.29%**	**46.25%**	**46.40%**
7	75.77%	52.54%	46.31%	45.94%	45.92%	46.07%
5	76.84%	52.07%	46.00%	45.67%	45.59%	45.80%
3	78.25%	52.22%	45.32%	44.84%	44.82%	45.10%
1	80.30%	52.12%	44.10%	43.58%	43.52%	43.86%
0.5	80.79%	52.06%	43.95%	43.32%	43.24%	43.61%
0.1	80.82%	51.97%	43.39%	42.78%	42.76%	43.01%

Table 15: Hyperparameter sweep of MMA on ResNet: evaluation based performance on CIFAR-10 against ℓ_∞ bounded adversarial perturbations ($\varepsilon = 8/255$).

Margin	Standard	FGSM	PGD20	PGD100	PGD200	PGD20-50
12	84.54%	46.11%	34.56%	33.89%	33.81%	34.12%
20	83.12%	**47.48%**	36.99%	36.41%	36.37%	36.59%
32	82.37%	<u>47.08</u>%	**37.26%**	**36.71%**	**36.66%**	**36.85%**
48	82.03%	46.77%	36.64%	36.02%	35.98%	36.22%
60	81.67%	46.79%	36.91%	36.35%	36.32%	36.51%
72	81.73%	46.40%	36.55%	36.01%	35.93%	36.21%

E.11 Adversarial Robustness of the Model Trained using FOAR with Different Initializations

FOAR achieves the best adversarial robustness using PGD1 initialization, so we only present this variation of FOAR in Section 5.

Table 16: Performance of FOAR with different initializations on CIFAR-10 against white-box and transfer-based black-box ℓ_∞ bounded adversarial perturbations ($\varepsilon = 8/255$).

Initialization	Standard	FGSM	PGD20	PGD100	PGD200	PGD1000	PGD20-50
zero	74.43%	33.39%	23.65%	22.83%	22.79%	23.23%	68.58%
rand	73.96%	33.62%	24.71%	23.96%	23.93%	24.37%	**69.04%**
PGD1	65.84%	**36.96%**	**32.28%**	**31.87%**	**31.89%**	**32.08%**	63.56%