EFFICIENT SEMI-SUPERVISED ADVERSARIAL TRAINING WITHOUT GUESSING LABELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Adversarial training has been proved to be the most effective defensive strategy to protect models from adversarial attacks. In the practical application scenario of adversarial training, we face not only labeled data, but also an enormous amount of unlabeled data. However, existing adversarial training methods are naturally targeting supervised learning problems. To adapt to semi-supervised learning problems, they need to estimate labels for unlabeled data in advance, which inevitably degenerates the performance of the learned model due to the bias on the estimation of labels for unlabeled data. To mitigate this degeneration, in this paper, we propose a new semi-supervised adversarial training framework via maximizing AUCs which is also a minimax problem but treats the unlabeled samples as both positive and negative ones, so that we do not need to guess the labels for unlabeled data. Unsurprisingly, the minimax problem can be solved via the traditional adversarial training algorithm by extending singly stochastic gradients to triply stochastic gradients, to adapt to the three (i.e. positive, negative, and unlabeled) data sources. To further accelerate the training procedure, we transform the minimax adversarial training problem into an equivalent minimization one based on the kernel perspective. For the minimization problem, we discuss scalable and efficient algorithms not only for deep neural networks but also for kernel support vector machines. Extensive experimental results show that our algorithms not only achieve better generalization performance against various adversarial attacks, but also enjoy efficiency and scalability when considered from the kernel perspective.

1 INTRODUCTION

Machine learning models have long been confirmed to be vulnerable to adversarial examples which are specially crafted data that can easily subvert the predictions of the models (Goodfellow et al., 2014; Carlini & Wagner, 2017; Biggio et al., 2012). Then many studies have been published with the aim of finding countermeasures to protect these learning models (Papernot et al., 2016; Shafahi et al., 2019). Among existing defensive strategies, adversarial training (Madry et al., 2017) is proved to be the most effective one (Athalye et al., 2018). Generally, it can be defined as a minimax problem (Wang et al., 2020), where the inner maximization simulates the behavior of attackers to construct the most aggressive adversarial examples, and the outer minimization is a typical process to train the model to minimize the internal loss. In 2018, Schmidt et al. proposed that the improvement of adversarial robustness requires more data than common training. However, following this suggestion can be difficult due to the cost of gathering additional data and obtaining high-quality labels. Thus, in order to avoid the heavy cost of collecting labeled data, we need to handle not only labeled data but also a large number of unlabeled data in practical application scenarios.

However, adversarial training is naturally designed for supervised learning scenarios, which means that it is only applicable to labeled samples of the form (x, y). This is because generating the perturbation δ for an adversarial example relies on the divergence between the true label y and the predicted value $f(x)$ as shown in the upper row of Fig. 1. If the true label y is missing, i.e., the target for adversarial attack is absent, the loss between the true label y and the predicted value $f(x)$ is undefined which is explicitly shown in the lower row of Fig. 1. In order to solve this problem, there exist several works (Carmon et al., 2019; Miyato et al., 2018; Uesato et al., 2019; Zhai et al., 2019) devoted to semi-supervised adversarial training. These works estimate labels for unlabeled data in advance, which inevitably degenerates the performance of the learned model due to the biased estimation of labels for unlabeled data.

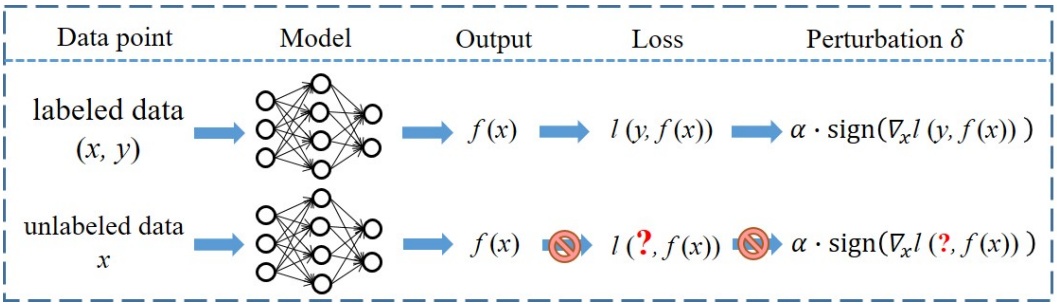


Figure 1: The challenge of generating perturbations for unlabeled data in the l_∞ norm.

Besides, the performance of adversarial training algorithms is mostly measured by accuracy. For a highly imbalanced dataset with two classes, a classifier that predicts all samples to belong to the dominant class will achieve high prediction accuracy, but have poor generalization performance since it will misclassify all samples in another class. Thus, accuracy is not a good metric for the imbalanced classification scenario. Instead, the area under the ROC curve (AUC) (Hanley & McNeil, 1982), which measures the probability of a randomly drawn positive sample to have a higher decision value than a randomly drawn negative sample, is a more meaningful metric for highly-imbalanced datasets.

To alleviate these problems, in this paper, we propose a new semi-supervised adversarial training framework via maximizing AUCs (S2AT-AUC) which can be formulated as a minimax problem. Specifically, this framework treats the unlabeled samples as both positive and negative ones, so that we do not need to guess the labels for unlabeled data, which leads to an unbiased estimation. Undoubtedly, the minimax problem can be solved via a strategy similar to the standard adversarial training algorithm, i.e., optimizing the model with a projected gradient descent (PGD) adversary (Madry et al., 2017). The difference is that we extend singly stochastic gradients to triply stochastic gradients to adapt to the three data sources (i.e., positive, negative and unlabeled data). However, the K-step PGD attack for generating adversarial examples is known to cost much. To further accelerate the training procedure, we transform the minimax adversarial training problem into an equivalent minimization one based on the kernel perspective via the connection between perturbations in the linear and kernel spaces. For the minimization problem, we discuss it not only on deep neural networks (DNNs) but also on kernel support vector machines (SVMs). Our main contributions are summarized as follows:

- We propose an ingenious strategy of adversarial training for semi-supervised learning, where we do not need to calculate a pseudo-label for the unlabeled samples, and we just treat them as both positive and negative ones.
- We propose an efficient semi-supervised adversarial training framework for nonlinear AUC maximization which can be applied to both DNNs and kernel SVMs. With extensive experimental results, we show its superiority.

2 Semi-Supervised Adversarial Training via AUC Maximization: S2AT-AUC

In this section, we first give a brief review of existing methods to generate adversarial examples for unlabeled data, then introduce our strategy on this issue for AUC optimization. Based on that, we propose our minimax semi-supervised adversarial training function for maximizing AUCs.

2.1 Challenge of Generating Adversarial Examples for Unlabeled Data

In this part, we extend the concept of adversarial examples from labeled samples to unlabeled ones and discuss the challenge of generating such adversarial examples. For a labeled sample (x, y), the generation of its adversarial example is normally formulated as the following maximization problem:

$$\max_{\|x'-x\|_2 \leq \epsilon} \quad l(y, f(x')), \tag{1}$$

where $x' = x + \delta$ is the adversarial example of x, δ is the perturbation and ϵ is the maximum perturbation radius. We focus on the l_2 norm constraint in this paper. As mentioned in Fig. 1,

the formulation (1) cannot be directly used for generating the adversarial example for an unlabeled sample due to the absence of label y.

To solve this problem, existing countermeasures can be roughly divided into two classes. One is using the predicted value $f(x)$ as its label, the other is generating a pseudo-label $\hat{y}$ as its label. For the former one, Miyato et al. (2018) proposed the VAT algorithm which evaluates the loss via computing the divergence between $f(x)$ and $f(x')$. However, in this case, the perturbation δ is unable to be computed normally as $\alpha \cdot g/\left\|g\right\|_2$, where $g = \nabla_{x'} l(f(x), f(x'))$, α is the step size of the PGD attack, since the loss l takes the minimal value at $\delta = 0$, so its first derivative at that point is 0 as well. To address this issue, they use the second-order Taylor expansion to approximate the value of δ. For the latter one, Carmon et al. (2019) and Uesato et al. (2019) use a well-known approach of semi-supervised learning called self-training (Rosenberg et al., 2005). This approach firstly trains an intermediate model with labeled samples, then uses it to generate pseudo-labels $\hat{y}$ for unlabeled samples.

However, since these methods need to guess labels, this will inevitably lead to the performance degradation of the learned models due to the biased estimation of labels for unlabeled data. To alleviate this problem, we introduce an unbiased estimation method in the following part.

2.2 Semi-Supervised AUC Optimization without Guessing Labels for Unlabeled Data

Let $\mathcal{D} = \mathcal{D}_p \cup \mathcal{D}_n$ denote the labeled dataset, where $\mathcal{D}_p = \{x_i^p\}_{i=1}^{N_p} \sim p_P(x)$, $\mathcal{D}_n = \{x_j^n\}_{j=1}^{N_n} \sim p_N(x)$, x_i^p, $x_j^n \in \mathbb{R}^d$, $p_P(x) = p(x|y=+1)$ and $p_N(x) = p(x|y=-1)$ are the distributions for positive and negative samples respectively[1]. In the semi-supervised setting, the unlabeled dataset D_u is considered to be drawn from a mixture of the positive and negative distributions, i.e., $D_u = \{x_i^u\}_{i=1}^{N_u} \sim p(x) = \theta_P p_P(x) + \theta_N p_N(x)$, where θ_P, θ_N are the prior probabilities of the positive and negative classes with $\theta_P + \theta_N = 1$.

Recently, Zheng & Ming (2018) proved that it is unnecessary to estimate class prior probabilities θ_P and θ_N to achieve unbiased semi-supervised AUC optimization. Instead, PU AUC risk R_{PU} and NU AUC risk R_{NU} can be equivalent to the supervised PN AUC risk R_{PN} with a linear transformation, i.e., $R_{PU} + R_{NU} - \frac{1}{2} = R_{PN}$, where PU AUC risk R_{PU} is estimated by the positive data and the unlabeled data regarded as negative data, and NU AUC risk R_{NU} is estimated by the negative data and the unlabeled data regarded as positive data. Specifically, R_{PN}, R_{PU} and R_{NU} are defined as follows.

$$R_{PN} = \mathbb{E}_{x^p \in \mathcal{D}_p(x)}[\mathbb{E}_{x^n \in \mathcal{D}_n(x)}[l(x^p, x^n)]], \tag{2}$$

$$R_{PU} = \mathbb{E}_{x^p \in \mathcal{D}_p(x)}[\mathbb{E}_{x^u \in \mathcal{D}_u(x)}[l(x^p, x^u)]], \tag{3}$$

$$R_{NU} = \mathbb{E}_{x^u \in \mathcal{D}_u(x)}[\mathbb{E}_{x^n \in \mathcal{D}_n(x)}[l(x^u, x^n)]]. \tag{4}$$

Importantly, the linear transformation $R_{PU} + R_{NU} - \frac{1}{2} = R_{PN}$ means that unbiased AUC risk estimation can be achieved without knowing θ_P and θ_N.

When a classifier is trained in practice, we use the empirical risks $\hat{R}$ instead of the expected risks R, i.e., $\hat{R}_{PN} = \frac{1}{N_p N_n} \sum_{i=1}^{N_p} \sum_{j=1}^{N_n} l(x_i^p, x_j^n)$. In this way, the semi-supervised AUC optimization can be formulated as

$$\hat{R}_{PNU} = \beta \hat{R}_{PN} + (1-\beta)\left(\hat{R}_{PU} + \hat{R}_{NU} - \frac{1}{2}\right), \tag{5}$$

where $\beta \in [0, 1]$ is the trade-off parameter. It should be noted that the loss function we use in this paper is the convex pairwise hinge loss $l = \max(0, 1 - f(x_i^p) + f(x_j^n))$. Other convex surrogate loss functions presented in (Zhiyuan et al., 2020) are applicable as well.

2.3 Framework of S2AT-AUC

Based on Eq. (5), we propose our framework of semi-supervised adversarial training for AUC optimization in the following. The inner maximization problem actually follows the principle

[1]Since binary classification is the basis of multi-class classification, in this paper, we discuss the binary classification problem. It should be noted that our algorithms can be easily extended to multi-class classification via one vs. one (OVO) or one vs. all (OVA) strategies (Duan et al., 2007).

of adversarial attack and aims to maximize the loss to construct the most aggressive adversarial examples. The goal of the outer minimization problem is to find parameters f by minimizing $\mathcal{L}(f; \mathcal{D}_p, \mathcal{D}_n, \mathcal{D}_u)$, the average internal pairwise loss caused by adversarial examples, defined as follows in detail.

$$\begin{aligned}\mathcal{L}(f; \mathcal{D}_p, \mathcal{D}_n, \mathcal{D}_u) \stackrel{\text{def}}{=} & \beta \frac{1}{N_p N_n} \sum_{i=1}^{N_p} \sum_{j=1}^{N_n} \max_{x_i'^p, x_j'^n} [1 - f(x_i'^p) + f(x_j'^n)]_+ \\ & + (1-\beta) \left\{ \frac{1}{N_p N_u} \sum_{i=1}^{N_p} \sum_{j=1}^{N_u} \max_{x_i'^p, x_j'^u} [1 - f(x_i'^p) + f(x_j'^u)]_+ \right. \\ & \left. + \frac{1}{N_u N_n} \sum_{i=1}^{N_u} \sum_{j=1}^{N_n} \max_{x_i'^u, x_j'^n} [1 - f(x_i'^u) + f(x_j'^n)]_+ - \frac{1}{2} \right\} \\ s.t. \quad & \forall\, x_i^p, x_j^n, x_k^u \in \mathcal{D}_p \times \mathcal{D}_n \times \mathcal{D}_u : \left\| x_i'^p - x_i^p \right\|_2 \leq \epsilon; \ \ \left\| x_j'^n - x_j^n \right\|_2 \leq \epsilon; \ \ \left\| x_k'^u - x_k^u \right\|_2 \leq \epsilon.\end{aligned} \tag{6}$$

where $[\pi]_+$ denotes $\max(0, \pi)$ which corresponds to the hinge loss.

Different from strategies to generate adversarial examples for unlabeled data introduced in Section 2.1, we simply regard the unlabeled sample as positive and negative ones at the same time. Thus, it is unnecessary to estimate the labels of unlabeled samples, which achieves unbiased AUC risk estimation. In this case, we have two adversarial examples for one unlabeled sample which correspond to positive and negative ones respectively (The intuitive explanation can be found in Fig. 2b).

2.4 S2AT-AUC ON MINIMAX PROBLEM

The objective function of S2AT-AUC on the minimax problem is shown in Eq. (6). We define $l_1 = [1 - f(x_i'^p) + f(x_j'^n)]_+$, $l_2 = [1 - f(x_i'^p) + f(x_j'^u)]_+$ and $l_3 = [1 - f(x_i'^u) + f(x_j'^n)]_+$. The detailed algorithm is summarized in Algorithm 1. Π_B in the algorithm is the projection operator on the perturbation set B and $B(x, \epsilon) = \{x + \delta \ s.t. \|\delta\|_2 \leq \epsilon\}$. The inner loop generates the most aggressive data that maximizes the loss via the PGD attack (Madry et al., 2017) (line 6-13), which perturbs natural data in the given perturbation boundary with a small step size α. The difference is that we extend the original singly stochastic gradients to triply stochastic gradients in order to adapt to the three data sources (x^p, x^n, x^u). The outer loop updates the model using optimizers such as stochastic gradient descent (SGD) (Bottou, 2010) and ADAM (Kingma & Ba, 2014).

Algorithm 1 S2AT-AUC on Minimax Problem

Input: $\mathcal{D}_p$, $\mathcal{D}_n$, $\mathcal{D}_u$: training sets, T: number of epochs, ϵ: maximum perturbation radius, γ: learning rate, K: PGD steps, α: PGD step size which is defaultly set as $\frac{2.5 \times \epsilon}{K}$.
Output: f.

1: **for** epoch$= 1, \cdots, T$ **do**
2: Sample x_i^p from $\mathcal{D}_p$.
3: Sample x_j^n from $\mathcal{D}_n$.
4: Sample x_t^u from $\mathcal{D}_u$.
5: $x_i'^p = x_i^p, x_j'^n = x_j^n, x_i'^u = x_j'^u = x_t^u$. *// Ready for PGD attack*
6: **for** $k = 1, \cdots, K$ **do**
7: $g_1 = \nabla_{x_i'^p} l_1(f, x_i'^p, x_j'^n)$; $g_2 = \nabla_{x_j'^n} l_1(f, x_i'^p, x_j'^n)$.
8: $g_3 = \nabla_{x_i'^p} l_2(f, x_i'^p, x_j'^u)$; $g_4 = \nabla_{x_j'^u} l_2(f, x_i'^p, x_j'^u)$.
9: $g_5 = \nabla_{x_i'^u} l_3(f, x_i'^u, x_j'^n)$; $g_6 = \nabla_{x_j'^n} l_3(f, x_i'^u, x_j'^n)$.
10: $x_i'^p = \Pi_{B(x_i^p, \epsilon)}(x_i'^p + \alpha \cdot g_1 / \|g_1\|_2)$; $x_j'^n = \Pi_{B(x_j^n, \epsilon)}(x_j'^n + \alpha \cdot g_2 / \|g_2\|_2)$.
11: $x_i'^p = \Pi_{B(x_i^p, \epsilon)}(x_i'^p + \alpha \cdot g_3 / \|g_3\|_2)$; $x_j'^u = \Pi_{B(x_t^u, \epsilon)}(x_j'^u + \alpha \cdot g_4 / \|g_4\|_2)$.
12: $x_i'^u = \Pi_{B(x_t^u, \epsilon)}(x_i'^u + \alpha \cdot g_5 / \|g_5\|_2)$; $x_j'^n = \Pi_{B(x_j^n, \epsilon)}(x_j'^n + \alpha \cdot g_6 / \|g_6\|_2)$.
13: **end for**
14: $f = f - \gamma \nabla_f \mathcal{L}(f; x_i^p, x_j^n, x_t^u)$. *// It can be replaced by other updating rule like ADAM.*
15: **end for**

3 S2AT-AUC ON THE KERNEL PERSPECTIVE

It is generally known that standard adversarial training with minimax formulation runs slowly due to the high cost of generating strong adversarial examples via K-step PGD attack, which makes it impractical on large-scale problems. Thus in this section, we propose a new semi-supervised adversarial strategy for nonlinear AUC optimization from the kernel perspective[2] which can transform the original minimax problem into a minimization one. In the following, we first build some primary results for the adversarial training from the kernel perspective, then discuss the detailed algorithms on DNNs and kernel SVMs respectively.

3.1 PRIMARY RESULTS FROM THE KERNEL PERSPECTIVE

Our kernelized semi-supervised AUC adversarial training is formulated as a minimax optimization problem as follows. Here a regularization term is added into Eq. (6) to reduce the risk of over-fitting:

$$\min_{f\in\mathcal{H}} \quad \frac{1}{2}\|f\|_{\mathcal{H}}^2 + \mathcal{L}(f;\mathcal{D}_p,\mathcal{D}_n,\mathcal{D}_u) \tag{7}$$

where f is the model function in the RKHS $\mathcal{H}$ and $\|\cdot\|_{\mathcal{H}}$ stands for the norm in $\mathcal{H}$. Note that $f(\cdot)$ can be written as $\langle f, \phi(\cdot)\rangle_{\mathcal{H}}$ and $\phi(\cdot)$ is the feature mapping. As shown in Figs. 2a and 2b, δ are the perturbations added to the data samples in the linear space, then we can see that a more complicated decision boundary is needed to separate them. Moreover, when the adversarial examples $x+\delta$ are mapped into the kernel space, $\phi(x+\delta)$ will become unpredictable like Fig. 2c, which significantly increases the difficulty of data processing and computation. Fortunately, Theorem 1 builds the relationship between perturbations in the linear and the kernel spaces.

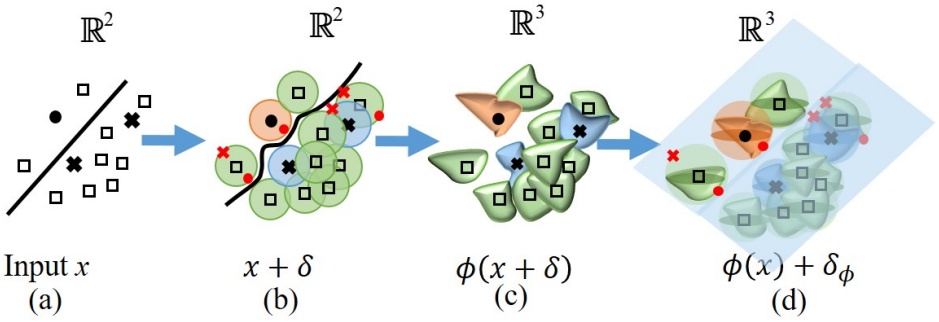


Figure 2: Conceptual illustration of perturbations in the linear and kernel spaces. (Here solid circles denote positive samples, solid crosses denote negative samples, hollow squares denote unlabeled samples, the red circle and cross are adversarial examples of positive and negative samples respectively. Note that one unlabeled sample has two adversarial examples.)

Theorem 1. *(Xu et al., 2009) Suppose the kernel function has the form $k(x,x') = f(\|x-x'\|)$, with $f:\mathbb{R}^+\to\mathbb{R}$, a decreasing function. Denote by $\mathcal{H}$ the RKHS space of $k(\cdot,\cdot)$ and $\phi(\cdot)$ the corresponding feature mapping. Then we have for any $x\in\mathbb{R}^n, w\in\mathcal{H}$ and ϵ>0,*

$$\sup_{\|\delta\|_2\le\epsilon} \langle w,\phi(x+\delta)\rangle_{\mathcal{H}} \le \sup_{\|\delta_\phi\|_{\mathcal{H}}\le\sqrt{2f(0)-2f(\epsilon)}} \langle w,\phi(x)+\delta_\phi\rangle_{\mathcal{H}}. \tag{8}$$

Since the perturbation range of $\phi(x)+\delta_\phi$ tightly covers that of $\phi(x+\delta)$, which is also intuitively shown in Fig. 2d, we apply $\phi(x)+\delta_\phi$ to deal with the following computation, thus the perturbations can be more tractable in the kernel space. Then the objective function (7) can be rewritten as

$$\min_{f\in\mathcal{H}} \quad \frac{1}{2}\|f\|_{\mathcal{H}}^2 + \beta\frac{1}{N_pN_n}\sum_{i=1}^{N_p}\sum_{j=1}^{N_n}\max_{\Phi(x_i^p),\Phi(x_j^n)}[1-\langle f,\Phi(x_i^p)\rangle_{\mathcal{H}}+\langle f,\Phi(x_j^n)\rangle_{\mathcal{H}}]_+$$

[2]The kernel perspective means that our function f is in the reproducing kernel Hilbert space (RKHS) (Iii, 2004)

$$+ (1-\beta)\left\{\frac{1}{N_pN_u}\sum_{i=1}^{N_p}\sum_{j=1}^{N_u}\max_{\Phi(x_i^p),\Phi(x_j^u)}[1-\langle f,\Phi(x_i^p)\rangle_{\mathcal{H}}+\langle f,\Phi(x_j^u)\rangle_{\mathcal{H}}]_+ \right. \tag{9}$$

$$\left. + \frac{1}{N_uN_n}\sum_{i=1}^{N_u}\sum_{j=1}^{N_n}\max_{\Phi(x_i^u),\Phi(x_j^n)}[1-\langle f,\Phi(x_i^u)\rangle_{\mathcal{H}}+\langle f,\Phi(x_j^n)\rangle_{\mathcal{H}}]_+ - \frac{1}{2}\right\}$$

$$s.t. \quad \forall\, x_i^p, x_j^n, x_k^u \in \mathcal{D}_p \times \mathcal{D}_n \times \mathcal{D}_u :$$

$$\|\Phi(x_i^p)-\phi(x_i^p)\|_2 \le \epsilon', \quad \left\|\Phi(x_j^n)-\phi(x_j^n)\right\|_2 \le \epsilon', \quad \|\Phi(x_k^u)-\phi(x_k^u)\|_2 \le \epsilon',$$

where $\Phi(x_i^p) = \phi(x_i^p) + \delta_{\phi i}^p$, $\Phi(x_j^n) = \phi(x_j^n) + \delta_{\phi j}^n$, $\Phi(x_k^u) = \phi(x_k^u) + \delta_{\phi k}^u$ and ϵ' is $\sqrt{2f(0)-2f(\epsilon)}$.

Since Eq. (9) is still a minimax problem that is hard to be solved after the step above, we propose the simplified and equivalent form of the inner maximization via the following theorem.

Theorem 2. *If f is a function in an RKHS $\mathcal{H}$, the inner maximization problem* $\max_{\Phi(x_i^p),\Phi(x_j^n)}[1-\langle f,\Phi(x_i^p)\rangle_{\mathcal{H}}+\langle f,\Phi(x_j^n)\rangle_{\mathcal{H}}]_+$ *is equivalent to the regularized loss function* $[1-f(x_i^p)+f(x_j^n)+2\epsilon'\|f\|_{\mathcal{H}}]_+$.

The detailed proof of Theorem 2 is provided in Appendix A.1. Following this theorem, the transformation of the other two terms in Eq. (9) can also be easily obtained. Thus, the minimax objective function (9) can be written as the following minimization problem:

$$\min_{f\in\mathcal{H}} \ \frac{1}{2}\|f\|_{\mathcal{H}}^2 + \beta\frac{1}{N_pN_n}\sum_{i=1}^{N_p}\sum_{j=1}^{N_n}[1-f(x_i^p)+f(x_j^n)+2\epsilon'\|f\|_{\mathcal{H}}]_+$$

$$+ (1-\beta)\left\{\frac{1}{N_pN_u}\sum_{i=1}^{N_p}\sum_{j=1}^{N_u}[1-f(x_i^p)+f(x_j^u)+2\epsilon'\|f\|_{\mathcal{H}}]_+ \right. \tag{10}$$

$$\left. + \frac{1}{N_uN_n}\sum_{i=1}^{N_u}\sum_{j=1}^{N_n}[1-f(x_i^u)+f(x_j^n)+2\epsilon'\|f\|_{\mathcal{H}}]_+\right\}.$$

Since the minimax formulation is transformed into a single-layer minimization one, the K-step PGD attack for generating adversarial examples can be skillfully escaped. Thus this minimization problem can be easily extended to large-scale datasets.

3.2 Specific Algorithms

Based on the minimization formulation of semi-supervised adversarial training for nonlinear AUC optimization (10), our S2AT-AUC can be applied to both DNNs and kernel SVMs as follows.

3.2.1 S2AT-AUC for DNNs

Since the RKHS norm $\|f\|_{\mathcal{H}}$ cannot be computed on DNNs, here we use the lower approximation of $\|f\|_{\mathcal{H}}$, which was proposed by Bietti et al. (2019):

$$\|f\|_{\mathcal{H}} \ge \|f\|_{\delta}^2 := \sup_{\|\delta\|_2\le 1} f(x+\delta) - f(x). \tag{11}$$

In this case, the minimization problem (10) can be solved directly by gradient descent optimization algorithms such as SGD and ADAM.

3.2.2 S2AT-AUC for Kernel SVMs

In this part, we efficiently solve the minimization problem (10) by applying the kernel-based semi-supervised AUC learning algorithm with quadruply stochastic gradients (QSG-S2AUC) (Shi et al., 2019), which is a powerful technique for scalable nonlinear AUC learning.

Specifically, QSG-S2AUC first uses the random Fourier feature method (RFF) (Rahimi & Recht, 2008) to approximate the kernel function instead of computing it directly, then uses the quadruply stochastic gradients w.r.t. the pairwise loss and random features to iteratively update the objective function. The detailed optimization procedure is provided in Appendix A.2.

4 EXPERIMENTS

In this section, we evaluate the performance of S2AT-AUC on DNNs and kernel SVMs in the semi-supervised learning scenario.

4.1 EXPERIMENTAL SETUP

Compared Algorithms. We compare the AUC performance of S2AT-AUC with the state-of-the-art semi-supervised adversarial training algorithms on DNNs and semi-supervised AUC maximization algorithms on SVMs as follows:

- **VAT**: (Miyato et al., 2018) A semi-supervised adversarial training on DNNs which treats the predicted value $f(x)$ as its label y and optimizes the accuracy metric.
- **UAT**: (Uesato et al., 2019) A semi-supervised adversarial training on DNNs which generates the pseudo labels via self-training and optimizes the accuracy metric.
- **S2AT-AUC(K)**: Our semi-supervised adversarial training algorithm for nonlinear AUC maximization on DNNs from the kernel perspective.
- **S2AT-AUC(M)**: Our semi-supervised adversarial training algorithm for nonlinear AUC maximization based on the minimax problem for DNNs.
- **PNU-AUC**: A kernel-based semi-supervised AUC optimization algorithm based on positive and unlabeled learning for SVMs (Sakai et al., 2017).
- **SAMULT**: A kernel-based semi-supervised AUC optimization method which achieves unbiased AUC risk estimation by treating unlabeled data as both positive and negative data for SVMs (Zheng & Ming, 2018).
- **QSG-S2AUC**: A kernelized scalable quadruply stochastic gradient algorithm for nonlinear semi-supervised AUC optimization on SVMs (Shi et al., 2019).
- **S2AT-AUC(S)**: Our scalable semi-supervised adversarial training algorithm for nonlinear AUC maximization on kernel SVMs.

It is notable that the former four algorithms work on DNNs, while the rest work on kernel SVMs.

Datasets. The experiments are conducted on large-scale datasets MNIST8m (Lecun & Bottou, 1998) and CIFAR10 (Krizhevsky & Hinton, 2009). Since we focus on binary classification, here we select two similar classes from the datasets respectively. Their dimensions and sample sizes are summarized in Table 1. For all datasets, we set the imbalanced ratio (N_n/N_p) as 10.0. Moreover, we also do the experiments on the high dimensional and highly imbalanced dataset Sector, whose imbalanced ratio is 148.12 (one vs. all).

Table 1: Datasets used in the experiments.

Datasets	Dimensions	Sizes
CIFAR10 automobile vs. truck	3,072	6,600
CIFAR10 dog vs. horse	3,072	6,600
MNIST8m 0 vs. 4	784	550,000
MNIST8m 6 vs. 8	784	550,000
Sector	55,197	6,412

Attack Settings. We use four commonly used adversarial attack methods to construct adversarial examples: FGSM (Goodfellow et al., 2014), PGD10 (PGD with 10 steps) (Madry et al., 2017), C&W (Carlini & Wagner, 2017) and ZOO (Chen et al., 2017), where the former three ones belong to white-box attack and the last one belongs to black-box attack. Although these attacks are initially proposed for DNNs, they are also applicable to other learning models.

All the attacks are performed with their l_2 version. For FGSM and PGD10, we set the maximum perturbation $\epsilon = 3$ for MNIST8m, $\epsilon = 1.5$ for CIFAR10 and $\epsilon = 1$ for Sector, the stepsize for PGD10 is $\epsilon/4$, which is a standard setting for adversarial attack (Madry et al., 2017; Ding et al., 2018). For ZOO, we use the ZOO-ADAM algorithm and set the stepsize $\eta = 0.01$, ADAM parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$.

Table 2: AUC performance with standard deviation on MNIST8m 0 vs. 4 against different attacks.

	Model	Clean	FGSM	PGD10	C&W	ZOO
DNNs	**VAT**	95.96±0.37	92.23±0.42	81.29±0.56	68.35±0.79	70.86±0.72
	UAT	94.45±0.26	90.33±0.36	84.84±0.52	70.55±0.89	71.74±0.73
	S2AT-AUC(K)	**99.93±0.04**	94.33±0.37	88.35±0.69	72.47±0.86	73.79±0.63
	S2AT-AUC(M)	99.05±0.11	**97.93±0.46**	**93.35±0.51**	**80.88±0.46**	**79.69±0.78**
SVMs	**PNU-AUC**	99.82±0.07	98.50±0.22	95.47±0.48	82.82±0.65	77.23±0.88
	SAMULT	99.19±0.12	97.76±0.31	95.28±0.59	82.98±0.79	77.96±0.62
	QSG-S2AUC	**99.85±0.06**	99.45±0.21	95.49±0.29	82.94±0.54	78.38±0.82
	S2AT-AUC(S)	99.84±0.27	**99.78±0.23**	**98.23±0.52**	**85.39±0.58**	**82.35±0.64**

Table 3: AUC performance with standard deviation on CIFAR dog vs. horse against different attacks.

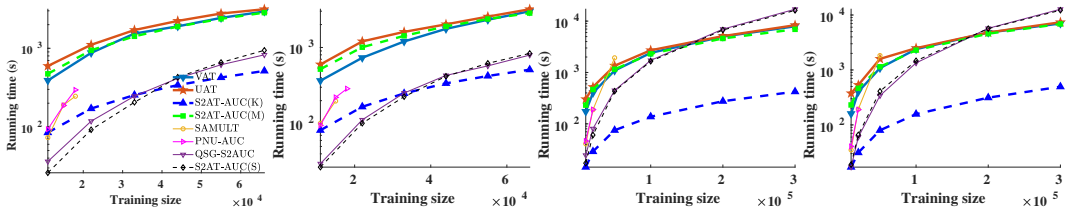
	Model	Clean	FGSM	PGD10	C&W	ZOO
DNNs	**VAT**	76.33±0.37	72.07±0.55	60.67±0.79	50.68±0.63	52.15±0.82
	UAT	76.68±0.31	70.59±0.62	64.44±0.82	56.33±0.74	58.66±0.91
	S2AT-AUC(K)	**84.13±0.39**	72.57±0.62	68.09±0.64	61.33±0.76	62.59±0.84
	S2AT-AUC(M)	81.80±0.49	**75.39±0.32**	**72.51±0.59**	**65.04±0.66**	**65.39±0.82**
SVMs	**PNU-AUC**	68.54±0.44	65.81±0.67	64.67±0.38	58.16±1.22	62.53±0.89
	SAMULT	68.81±0.54	66.37±0.53	65.66±0.67	59.90±0.89	62.99±1.14
	QSG-S2AUC	**69.92±0.59**	66.57±0.77	65.91±0.74	60.28±0.82	62.86±0.94
	S2AT-AUC(S)	69.51±0.39	**68.05±0.61**	**67.18±0.68**	**63.37±0.89**	**65.05±0.76**

Implementation. All the experiments are conducted on a PC with 48 2.2GHz cores, 80GB RAM and four Nvidia 1080ti GPUs. The kernel function that we use for algorithms on SVMs is the RBF kernel $k(x, x') = \exp(-\sigma\|x - x'\|_2^2)$. The hyper-parameter σ is chosen via cross-validation, searching in the region $\{\sigma | -3 \leq \log_2 \sigma \leq 3\}$. For algorithms on DNNs, we use the PreAct ResNet18 architecture for CIFAR10 and use two convolutional networks with 16 and 32 convolutional filters followed by a fully connected layer of 100 units for MNIST8m, which are the same models as provided by Wong et al. (2020). The trade-off parameter β is searched from 0 to 1 at intervals of 0.1. Since the algorithms are all under l_2-norm constrained perturbations, we set $\epsilon = 3$ for MNIST8m, $\epsilon = 1.5$ for CIFAR10, $\epsilon = 1$ for Sector, and the step size is set as $\epsilon/4$.

4.2 Experimental Results

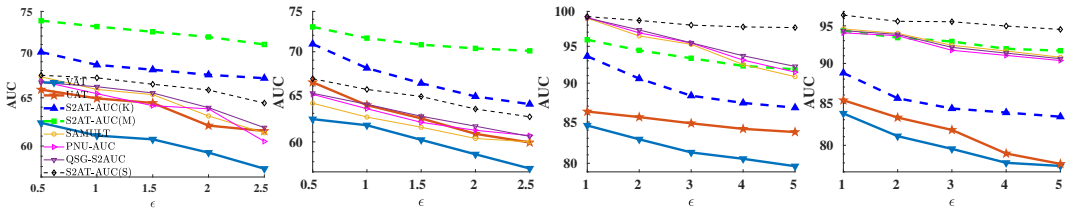
Comparison of AUC Performance with Existing Works. Firstly, we evaluate the AUC performance of these algorithms on clean datasets and adversarial examples generated by 4 attack methods. Due to the page limit, we only show the results of CIFAR10 dog vs. horse and MNIST8m 0 vs. 4, the results of other datasets are provided in Appendix A.3. Tables 2 and 3 clearly show that for algorithms on DNNs, our S2AT-AUC is much more effective when dealing with imbalanced data and also remains robust against different attacks compared with VAT and UAT since we optimize the AUC metric rather than the accuracy metric. For algorithms on SVMs, natural semi-supervised AUC optimization methods (PNU-AUC, SAMULT and QSG-S2AUC) are not robust against various adversarial examples. Our S2AT-AUC enjoys less superiority on clean data but achieves the best performance when defending against these attacks since standard generalization is at odds with robustness (Tsipras et al., 2018).

Comparison of Running Time with Different Sizes of Training Samples. Fig. 3 shows the running time of various algorithms when training samples with different sizes. We can find that when training on DNNs, S2AT-AUC(K) is much more efficient due to its one-layer objective function. For other algorithms on DNNs, the time-consuming factor lies in the K-step PGD attack. When training on SVMs, it is clear that PNU-AUC and SAMULT are time-consuming and even out of memory when training on large-scale datasets, while QSG-S2AUC and S2AT-AUC(S) enjoy high efficiency and require low memory which mainly benefits from the quadruply stochastic gradient algorithm. In general, our S2AT-AUC on the kernel perspective enjoy high scalability on large-scale datasets.

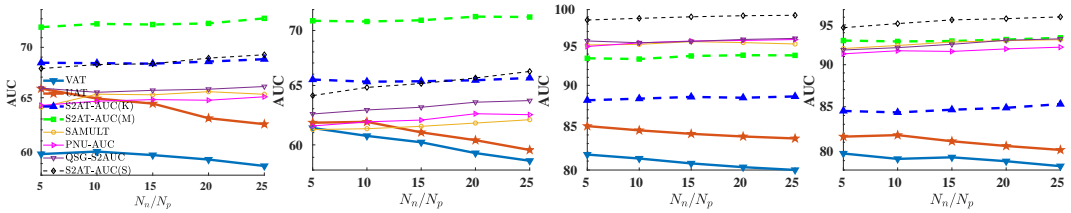


(a) CIFAR dog vs. horse (b) automobile vs. truck (c) MNIST8m 0 vs. 4 (d) MNIST8m 6 vs. 8

Figure 3: The running time of algorithms when training different sizes of samples. (The lines of SAMULT and PNU-AUC are incomplete because their implementations crash on large training sets.)



(a) CIFAR dog vs. horse (b) automobile vs. truck (c) MNIST8m 0 vs. 4 (d) MNIST8m 6 vs. 8



(e) CIFAR dog vs. horse (f) automobile vs. truck (g) MNIST8m 0 vs. 4 (h) MNIST8m 6 vs. 8

Figure 4: Sensitivity analysis of the maximum perturbation ϵ (Figs. 4a- 4d) and imbalanced ratio N_n/N_p (Figs. 4e-4h).

Sensitivity Analysis. We investigate the sensitivity of the algorithms against two parameters when evaluating their performance on test sets, which include the maximum perturbation radius ϵ for PGD attack and the imbalanced ratio N_n/N_p. For all experiments, the perturbed testing samples are generated by PGD10. The results are shown in Fig. 4.

The value of ϵ affects the attack power of adversarial examples. With the growth of ϵ, more perturbations will be added to the samples. Thus it is more challenging for the algorithms to make the correct classification. Nevertheless, our algorithm can still maintain superiority in the case of strong attacks, which can be seen clearly in Figs. 4a-4d.

The imbalanced ratio N_n/N_p measures the proportion of positive and negative samples in one dataset. Figs. 4e-4h show that algorithms which optimize the accuracy metric (VAT and UAT) have poor performance on datasets with high imbalanced ratio, while the AUC optimization algorithms can keep relatively stable, which demonstrates the superiority of AUC optimization algorithms on highly imbalanced datasets.

5 Conclusion

In this paper, we propose a new semi-supervised adversarial training strategy, S2AT-AUC, which is a framework for nonlinear AUC optimization that can be applied on both DNNs and kernel SVMs. Since we regard the unlabeled sample as both positive and negative ones which avoids guessing labels, an unbiased estimation can be achieved. Comprehensive experimental results verify that our framework achieves better generalization performance against various attacks when dealing with highly imbalanced datasets compared with existing algorithms. Moreover, it also enjoys high efficiency and scalability when considered from the kernel perspective.

REFERENCES

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

Alberto Bietti, Grégoire Mialon, Dexiong Chen, and Julien Mairal. A kernel perspective for regularizing deep neural networks. In *International Conference on Machine Learning*, pp. 664–674, 2019.

Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *International conference on machine learning*, pp. 1467–1474, 2012.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pp. 177–186. Springer, 2010.

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

Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, Percy Liang, and John C Duchi. Unlabeled data improves adversarial robustness. *Advances in Neural Information Processing Systems*, 2019.

Pinyu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Chojui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. *arXiv: Machine Learning*, pp. 15–26, 2017.

Bo Dai, Bo Xie, Niao He, Yingyu Liang, Anant Raj, Maria-Florina F Balcan, and Le Song. Scalable kernel methods via doubly stochastic gradients. In *NeurIPS*, pp. 3041–3049, 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.

Kai-Bo Duan, Jagath C Rajapakse, and Minh N Nguyen. One-versus-one and one-versus-all multiclass svm-rfe for gene selection in cancer classification. In *European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, pp. 47–56. Springer, 2007.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *In ICLR*, 2014.

James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

Hal Daumé Iii. From zero to reproducing kernel hilbert spaces in twelve pages or less. http://legacydirs.umiacs.umd.edu/ hal/docs/daume04rkhs.pdf, 2004.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

Y Lecun and L Bottou. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *International conference on learning representations*, 2017.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018.

Nicolas Papernot, Patrick Mcdaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. *IEEE symposium on security and privacy*, pp. 582–597, 2016.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pp. 1177–1184, 2008.

Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. Semi-supervised self-training of object detection models. 2005.

Tomoya Sakai, Marthinus Christoffel du Plessis, Gang Niu, and Masashi Sugiyama. Semi-supervised classification based on classification from positive and unlabeled data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2998–3006. JMLR. org, 2017.

Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. *Advances in Neural Information Processing Systems*, 2018.

Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John P Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free. pp. 3358–3369, 2019.

Wanli Shi, Bin Gu, Xiang Li, Xiang Geng, and Heng Huang. Quadruply stochastic gradients for large scale nonlinear semi-supervised auc optimization. In *Twenty-Eighth International Joint Conference on Artificial Intelligence IJCAI-19*, 2019.

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

Jonathan Uesato, Jean-Baptiste Alayrac, Po-Sen Huang, Robert Stanforth, Alhussein Fawzi, and Pushmeet Kohli. Are labels required for improving adversarial robustness? *arXiv preprint arXiv:1905.13725*, 2019.

Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. *international conference on learning representations*, 2020.

Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020.

Huan Xu, Constantine Caramanis, and Shie Mannor. Robustness and regularization of support vector machines. *JMLR*, 10:1485–1510, 2009.

Runtian Zhai, Tianle Cai, Di He, Chen Dan, Kun He, John Hopcroft, and Liwei Wang. Adversarially robust generalization just requires more unlabeled data. *Advances in Neural Information Processing Systems*, 2019.

Xie Zheng and Li Ming. Semi-supervised auc optimization without guessing labels of unlabeled data. *AAAI 2018*, 2018.

Dang Zhiyuan, Li Xiang, Deng Bin, Gu abd Cheng, and Huang Heng. Large-scale nonlinear auc maximization via triply stochastic gradients. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

A Appendix

A.1 Proof of Theorem 2

Since $\Phi(x) = \phi(x) + \delta_\phi$, the constraint can also be write as $\|\delta_\phi\|_2 \leq \epsilon'$, let $\mathcal{T} = \{\delta_\phi \mid \|\delta_\phi\|_2 \leq \epsilon'\}$.

We define $v = [1 - f(x_i^p) + f(x_j^n) + 2\epsilon' \|f\|_{\mathcal{H}}]_+$. To prove the theorem, we first prove $v \leq \max[1 - \langle f, \Phi(x_i^p)\rangle + \langle f, \Phi(x_j^n)\rangle]_+$, and then prove $\max[1 - \langle f, \Phi(x_i^p)\rangle + \langle f, \Phi(x_j^n)\rangle]_+ \leq v$. In the following, we give the details to prove these two sub-conclusions.

Step 1: We first prove $v \leq \max[1 - \langle f, \Phi(x_i^p)\rangle_{\mathcal{H}} + \langle f, \Phi(x_j^n)\rangle_{\mathcal{H}}]_+$.

Since, $\mathcal{T} = \{\delta_\phi | \|\delta_\phi\|_2 \leq \epsilon'\}$, we define two subsets of $\mathcal{T}$ as $\mathcal{T}_1' = \{-\epsilon' \frac{f}{\|f\|_{\mathcal{H}}}\}$, $\mathcal{T}_2' = \{\epsilon' \frac{f}{\|f\|_{\mathcal{H}}}\}$. Hence,

$$\begin{aligned}
&\max_{\delta_{\phi i}^p \in \mathcal{T}_1', \delta_{\phi j}^n \in \mathcal{T}_2'} [1 - \langle f, \phi(x_i^p) + \delta_{\phi i}^p\rangle_{\mathcal{H}} + \langle f, \phi(x_j^n) + \delta_{\phi j}^n\rangle_{\mathcal{H}}]_+ \\
=&\max_{\delta_{\phi i}^p \in \mathcal{T}_1', \delta_{\phi j}^n \in \mathcal{T}_2'} [1 - \langle f, \phi(x_i^p)\rangle_{\mathcal{H}} - \langle f, \delta_{\phi i}^p\rangle_{\mathcal{H}} + \langle f, \phi(x_j^n)\rangle_{\mathcal{H}} + \langle f, \delta_{\phi j}^n\rangle_{\mathcal{H}}]_+ \\
=&[1 - f(x_i^p) + f(x_j^n) + 2\epsilon' \|f\|_{\mathcal{H}}]_+.
\end{aligned}$$

Since $\mathcal{T}_1' \subseteq \mathcal{T}$, $\mathcal{T}_2' \subseteq \mathcal{T}$, the first sub-conclusion can be proved.

Step 2: Next we prove $\max[1 - \langle f, \Phi(x_i^p)\rangle_{\mathcal{H}} + \langle f, \Phi(x_j^n)\rangle_{\mathcal{H}}]_+ \leq v$.

$$\begin{aligned}
&\max_{\delta_{\phi i}^p, \delta_{\phi j}^n \in \mathcal{T}} [1 - \langle f, \phi(x_i^p) + \delta_{\phi i}^p\rangle_{\mathcal{H}} + \langle f, \phi(x_j^n) + \delta_{\phi j}^n\rangle_{\mathcal{H}}]_+ \\
=&\max_{\delta_{\phi i}^p, \delta_{\phi j}^n \in \mathcal{T}} [1 - \langle f, \phi(x_i^p)\rangle_{\mathcal{H}} - \langle f, \delta_{\phi i}^n\rangle_{\mathcal{H}} + \langle f, \phi(x_j^n)\rangle_{\mathcal{H}} + \langle f, \delta_{\phi j}^n\rangle_{\mathcal{H}}]_+ \\
\leq&\max_{\delta_{\phi i}^p, \delta_{\phi j}^n \in \mathcal{T}} [1 - \langle f, \phi(x_i^p)\rangle_{\mathcal{H}} + \|f\|_{\mathcal{H}} \cdot \|\delta_{\phi i}^p\|_2 + \langle f, \phi(x_j^n)\rangle_{\mathcal{H}} + \|f\|_{\mathcal{H}} \cdot \|\delta_{\phi j}^n\|_2]_+ \\
\leq&[1 - f(x_i^p) + f(x_j^n) + 2\epsilon' \|f\|_{\mathcal{H}}]_+.
\end{aligned}$$

The first inequality is due to the Cauchy-Schwarz inequality. The second inequality holds since $\|\delta_\phi\|_2 \leq \epsilon'$. Hence the second sub-conclusion holds.

Step 3: Combining these two steps, we have (12):

$$\max_{\substack{\|\Phi(x_i^p) - \phi(x_i^p)\|_2 \leq \epsilon', \\ \|\Phi(x_i^n) - \phi(x_i^n)\|_2 \leq \epsilon'}} [1 - \langle f, \Phi(x_i^p)\rangle_{\mathcal{H}} + \langle f, \Phi(x_j^n)\rangle_{\mathcal{H}}]_+ = [1 - f(x_i^p) + f(x_j^n) + 2\epsilon' \|f\|_{\mathcal{H}}]_+. \tag{12}$$

A.2 Optimization Procedure of S2AT-AUC for Kernel SVMs

In this part, we aim to efficiently solve the objective minimization problem (10) via QSG-S2AUC. In general, we first derive quadruply stochastic gradients for Eq. (10), and then provide an update rule to efficiently update the solution.

It should noted that the gradient of $[1 - f(x_i^p) + f(x_j^n) + 2\epsilon' \|f\|_{\mathcal{H}}]$ is $\mathbf{I}((1 - f(x_i^p) + f(x_j^n) + 2\epsilon' \|f\|_{\mathcal{H}})>0) \times (k(x_j^n, \cdot) - k(x_i^p, \cdot) + 2\epsilon' f(\cdot)/\|f\|_{\mathcal{H}})$, $\mathbf{I}(\pi)$ is the indicator function that equals 1 when π is true and 0 otherwise. For simplicity, we abbreviate $\mathbf{I}((1-f(x_i^p)+f(x_j^n)+2\epsilon' \|f\|_{\mathcal{H}})>0)$, $\mathbf{I}((1 - f(x_i^p) + f(x_j^u) + 2\epsilon' \|f\|_{\mathcal{H}})>0)$ and $\mathbf{I}((1 - f(x_i^u) + f(x_j^n) + 2\epsilon' \|f\|_{\mathcal{H}})>0)$ as I_1, I_2 and I_3 respectively.

Another important concept here is the RKHS. $\mathcal{H}$ is an RKHS if and only if there exists a $k(x, x')$, such that $\forall x \in \mathcal{X},\ k(x, \cdot) \in \mathcal{H}$ and $\forall f \in \mathcal{H}$, $\langle f(\cdot), k(x, \cdot)\rangle_{\mathcal{H}} = f(x)$. Applying the definition, we have $\nabla f(x) = k(x, \cdot)$ and $\nabla \|f\|_{\mathcal{H}}^2 = 2f$.

Here we only introduce the case that the loss is greater than 0 for the convenience of discussion.

Firstly, we randomly sample a positive sample x^p, a negative sample x^n and an unlabeled sample x^u from D_p, D_n and D_u respectively, then the gradient of Eq. (10) can be easily written as

$$\begin{aligned}
\nabla_f \mathcal{R} =& f + \beta I_1 (k(x^n, \cdot) - k(x^p, \cdot) + 2\epsilon' \frac{f(\cdot)}{\|f\|_{\mathcal{H}}}) \\
&+ (1 - \beta) \left\{ I_2 (k(x^u, \cdot) - k(x^p, \cdot) + 2\epsilon' \frac{f(\cdot)}{\|f\|_{\mathcal{H}}}) + I_3 (k(x^n, \cdot) - k(x^u, \cdot) + 2\epsilon' \frac{f(\cdot)}{\|f\|_{\mathcal{H}}}) \right\}
\end{aligned} \tag{13}$$

Since high computational complexity is still needed for kernel functions, we use the random feature approximation method Rahimi & Recht (2008) to approximate the stationary kernels such as RBF, Laplacian and Cauchy kernels by explicitly computing random features $\phi_\omega = \frac{1}{\sqrt{m}}[\phi_{\omega_1}(x), \phi_{\omega_2}(x), \cdots, \phi_{\omega_m}(x)]$, i.e., $k(x, x') \approx \phi_\omega(x)\phi_\omega^T(x')$, where m is the number of random features, $\phi_{\omega_i}(x)$ denotes $[\cos(\omega_i^T x), \sin(\omega_i^T x)]^T$ and ω is drawn from the measure $p(\omega)$. (The

detailed measure $p(\omega)$ of the kernels is shown in Table 1 of Dai et al. (2014).) Thus, Eq. (13) can be approximated as follows:

$$\begin{aligned}\nabla_f \hat{\mathcal{R}} =& f + \beta I_1(\phi_\omega(x^n)\phi_\omega(\cdot) - \phi_\omega(x^p)\phi_\omega(\cdot) + 2\epsilon' \frac{f(\cdot)}{\|f\|_{\mathcal{H}}}) \\ &+ (1-\beta)\Big\{ I_2(\phi_\omega(x^u)\phi_\omega(\cdot) - \phi_\omega(x^p)\phi_\omega(\cdot) + 2\epsilon' \frac{f(\cdot)}{\|f\|_{\mathcal{H}}}) \\ &+ I_3(\phi_\omega(x^n)\phi_\omega(\cdot) - \phi_\omega(x^u)\phi_\omega(\cdot) + 2\epsilon' \frac{f(\cdot)}{\|f\|_{\mathcal{H}}})\Big\}\end{aligned} \tag{14}$$

Since four sources of randomness (x^p, x^n, x^u and ω) are involved in $\nabla_f \hat{\mathcal{R}}$, we call it as quadruply stochastic functional gradients. Thus we can get the update rule for S2AT-AUC according to the principle of the SGD method.

$$f_{t+1}(\cdot) = f_t(\cdot) - \gamma_t \nabla_f \hat{\mathcal{R}} = \sum_{i=1}^{t} a_t^i \zeta_i(\cdot) \tag{15}$$

where γ_t is the stepsize in the t-th iteration, the initial value $f_1(\cdot) = 0$, the value of a_t^i[3] can be inferred as

$$a_t^i = -\gamma_i \prod_{j=i+1}^{t} \left[1 - \gamma_j \left(1 + \frac{2\epsilon'}{\|f_j\|_{\mathcal{H}}} (\beta I_1 + (1-\beta)(I_2 + I_3))\right)\right], \tag{16}$$

and

$$\begin{aligned}\zeta_i(\cdot) =& \beta I_1(\phi_\omega(x_i^n)\phi_\omega(\cdot) - \phi_\omega(x_i^p)\phi_\omega(\cdot)) \\ &+ (1-\beta)\left[I_2(\phi_\omega(x_i^u)\phi_\omega(\cdot) - \phi_\omega(x_i^p)\phi_\omega(\cdot))\right. \\ &\left.+ I_3(\phi_\omega(x_i^n)\phi_\omega(\cdot) - \phi_\omega(x_i^u)\phi_\omega(\cdot))\right].\end{aligned}$$

If we skip the step of random feature approximation and compute the kernel functions directly, the update rule will become

$$h_{t+1}(\cdot) = h_t(\cdot) - \gamma_t \nabla_f \mathcal{R} = \sum_{i=1}^{t} a_t^i \xi_i(\cdot) \tag{17}$$

where $\xi_i = \beta I_1(k(x_i^n, \cdot) - k(x_i^p, \cdot)) + (1-\beta)[I_2(k(x_i^u, \cdot) - k(x_i^p, \cdot)) + I_3(k(x_i^n, \cdot) - k(x_i^u, \cdot))]$.

Following the update rule (15), we present the training and prediction algorithms of S2AT-AUC on kernel SVM in Algorithm 2 and 3 respectively. Algorithm 2 performs data sampling, random feature sampling and maintains a collection of $\{a_i\}$, which is efficient in computation. An essential step here is sampling the random processes ω_i with seed i. Since the seeds keep aligned for the training and prediction processes in the same iteration, we only need to save the seeds instead of all the random features, which is memory friendly.

Algorithm 2 $\{\alpha_i\}_{i=1}^t = \textbf{Train}(\mathbb{P}(x, y))$

Input: $\mathcal{D}_p,\ \mathcal{D}_n,\ \mathcal{D}_u,\ p(\omega),\ \beta,\ \gamma$.

for $i = 1, \cdots, t$ **do**
  Sample $x^p \sim \mathcal{D}_p$.
  Sample $x^n \sim \mathcal{D}_n$.
  Sample $x^u \sim \mathcal{D}_u$.
  Sample $\omega_i \sim p(\omega)$ with seed i;
  $f(x) = \textbf{Predict}(x, \{\alpha_j\}_{j=1}^{i-1})$.
  $\alpha_i = -\gamma_i(\beta\,[I_1(\phi_{\omega_i}(x^n) - \phi_{\omega_i}(x^p))] + (1-\beta)[I_2(\phi_{\omega_i}(x^u) - \phi_{\omega_i}(x^p)) + I_3(\phi_{\omega_i}(x^n) - \phi_{\omega_i}(x^u))])$.
  $\alpha_j = (1 - \gamma_j(1 + \frac{2\epsilon'}{\|f_j\|_{\mathcal{H}}})[\beta I_1 + (1-\beta)(I_2 + I_3)])\alpha_j$ for $j = 1, \cdots, i-1$
end for

Algorithm 3 $f(x) = \textbf{Predict}(x, \{\alpha_i\}_{i=1}^t)$

Input: $p(\omega), \phi_\omega(x)$.

Set $f(x) = 0$.
for $i = 1, \cdots, t$ **do**
  Sample $\omega_i \sim p(\omega)$ with seed i;
  $f(x) = f(x) + \alpha_i \phi_{\omega_i}(x)$.
end for

[3]The value of a_t^i is gotten by expanding the middle term of Eq. (15) iteratively with the definition of $\nabla_f \hat{R}(f)$.

A.3 EXPERIMENTAL RESULTS

We show the AUC performance of the compared algorithms against various attacks on MNIST8m 6 vs. 8, CIFAR10 automobile vs. truck and high dimensional and highly imbalanced dataset Sector in Tables 4, 5, 6 respectively. It can be seen clearly that the conclusions we can get from the experimental results are consistent with those on CIFAR10 dog vs. horse and MNIST8m 0 vs. 4. Moreover, according to Table 6, our algorithm enjoys adversarial robustness on highly imbalanced and high dimensional datasets even when attacked by strong attacks like C&W and ZOO.

Table 4: AUC performance with standard deviation on MNIST8m 6 vs. 8 against different attacks.

	Model	Clean	FGSM	PGD10	C&W	ZOO
DNNs	**VAT**	94.03±0.44	89.96±0.52	79.59±0.82	63.28±0.61	61.57±0.83
	UAT	93.55±0.46	88.35±0.73	81.79±0.69	66.37±0.90	66.94±0.66
	S2AT-AUC(K)	**99.29±0.26**	92.69±0.47	84.35±0.68	70.35±0.88	71.22±0.72
	S2AT-AUC(M)	98.55±0.34	**96.73±0.72**	**92.89±0.65**	**79.37±0.59**	**78.44±0.82**
SVMs	**PNU-AUC**	98.19±0.34	96.79±0.52	91.72±0.89	69.64±0.93	71.26±0.81
	SAMULT	98.38±0.69	97.39±0.73	92.38±0.65	69.54±0.98	70.74±1.32
	QSG-S2AUC	98.90±0.38	97.01±0.41	92.11±0.73	72.87±0.86	73.39±0.94
	S2AT-AUC(S)	**98.99±0.41**	**98.55±0.59**	**95.92±0.98**	**76.20±0.79**	**77.79±0.67**

Table 5: AUC performance with standard deviation on CIFAR10 automobile vs. truck against different attacks.

	Model	Clean	FGSM	PGD10	C&W	ZOO
DNNs	**VAT**	73.10±0.46	69.54±0.72	61.21±0.59	50.42±0.33	51.81±0.76
	UAT	71.69±0.79	68.33±0.82	62.46±0.47	55.69±0.63	55.37±0.52
	S2AT-AUC(K)	**79.36±0.32**	70.59±0.58	65.33±0.62	60.48±0.84	61.09±0.79
	S2AT-AUC(M)	78.55±0.29	**74.67±0.69**	**70.85±0.82**	**64.96±0.79**	**65.35±0.62**
SVMs	**PNU-AUC**	67.82±0.28	65.86±0.41	62.04±0.47	60.29±0.94	58.85±0.83
	SAMULT	67.27±0.65	66.14±0.82	61.52±0.91	59.92±0.79	60.74±0.72
	QSG-S2AUC	**68.79±0.43**	66.98±0.69	62.69±0.82	59.34±0.74	59.80±0.91
	S2AT-AUC(S)	68.52±0.52	**67.67±0.72**	**64.71±0.69**	**63.94±0.82**	**63.07±0.96**

Table 6: AUC performance with standard deviation on Sector against different attacks.

	Model	Clean	FGSM	PGD10	C&W	ZOO
DNNs	**VAT**	85.76±0.33	79.95±0.59	71.46±0.73	57.27±0.64	55.79±0.74
	UAT	84.23±0.43	79.64±0.35	72.35±0.63	60.48±0.52	58.66±0.82
	S2AT-AUC(K)	**93.25±0.62**	83.75±0.88	77.32±0.94	67.59±0.87	66.06±0.74
	S2AT-AUC(M)	91.77±0.49	**85.46±0.75**	**80.88±0.86**	**73.02±0.69**	**72.34±0.72**
SVMs	**PNU-AUC**	80.89±0.57	77.44±0.62	73.07±0.83	61.59±0.79	64.10±0.94
	SAMULT	80.41±0.33	77.90±0.54	74.01±0.69	61.67±0.94	65.64±0.73
	QSG-S2AUC	**81.70±0.35**	79.52±0.47	74.94±0.64	64.16±0.59	66.79±0.87
	S2AT-AUC(S)	81.33±0.64	**80.64±0.39**	**77.35±0.72**	**67.72±0.54**	**69.29±0.88**