A Saddle-Point Dynamical System Approach for Robust Deep Learning

Yasaman Esfandiari[1], Keivan Ebrahimi[1], Aditya Balu[1], Nicola Elia[3], Umesh Vaidya[1,2], and Soumik Sarkar[1]

[1]Iowa State University
[2]Clemson University
[3]University of Minnesota

Abstract

We propose a novel discrete-time dynamical system-based framework for achieving adversarial robustness in machine learning models. Our algorithm is originated from robust optimization, which aims to find the saddle point of a min-max optimization problem in the presence of uncertainties. The robust learning problem is formulated as a robust optimization problem, and we introduce a discrete-time algorithm based on a saddle-point dynamical system (SDS) to solve this problem. Under the assumptions that the cost function is convex and uncertainties enter concavely in the robust learning problem, we analytically show that using a diminishing step-size, the stochastic version of our algorithm, SSDS converges asymptotically to the robust optimal solution. The algorithm is deployed for the training of adversarially robust deep neural networks. Although such training involves highly non-convex non-concave robust optimization problems, empirical results show that the algorithm can achieve significant robustness for deep learning. We compare the performance of our SSDS model to other state-of-the-art robust models, e.g., trained using the projected gradient descent (PGD)-training approach. From the empirical results, we find that SSDS training is computationally inexpensive (compared to PGD-training) while achieving comparable performances. SSDS training also helps robust models to maintain a relatively high level of performance for clean data as well as under black-box attacks.

Introduction

The success of adversarial perturbations to input data for deep learning models poses a significant challenge for the machine learning community. The need becomes safety- and life-critical, considering the application of deep learning based perception system for self-driving cars or security applications[5,25]. While pure white-box attacks[5,8,13,17,19,27] (where an adversary has full knowledge of the machine learning model) could be difficult to execute in practice, researchers have shown strong transferability of attacks[21] that can still cause significant damage.

As attacks became more and more powerful, several defense strategies have also been proposed. A popular category of defense strategy is adversarial training, where adversarial examples are added to the training set followed by training the network using the augmented dataset[13,27]. However, such methods seem to be quite sensitive to adversarial budget used for generating the adversarial examples as well as other training hyper-parameters. A more powerful and stable defense mechanism stems from decoupling the min-max robust optimization problem[18] related to robust learning using the Danskin's theorem[10]. Here, the inner maximization refers to finding the adversarial perturbation that would maximize the training loss. On the other hand, the outer maximization deals with minimization of the training loss for the perturbed inputs. The decoupling process leads to the class of algorithms where, at a training epoch, one can find the worst-case attacks concerning the current model. Then a model parameter update step is executed following the traditional training process using perturbed training set with the worst-case attacks. However, finding the worst-case perturbation for deep learning models is quite non-trivial and cannot be guaranteed primarily due to the highly non-convex nature of the cost surface. Typically, powerful attacks such as fast gradient sign method (FGSM)[13], Carlini-Wagner (CW)[8] and projected gradient descent (PGD)[18] are run in order to find the worst-case perturbations at every training epoch. However, it is observed empirically that the attacks with higher computational budgets seem to be more successful in approximating the

worst-case perturbations, e.g., 20 step PGD is much stronger than a single step PGD. Therefore, it is usually quite expensive computationally to find a robust deep learning model. Besides, there still remains a significant gap in the literature, in crafting theoretically sound algorithms for robust learning. Apart from a few studies, e.g., by Shaham *et al*. [23] that proposed a framework to justify the performance of adversarial training theoretically, this area has not been explored sufficiently.

In this paper, we introduce a new algorithm for adversarial training of Deep Neural Networks (DNN) based on continuous and discrete-time saddle point dynamical system introduced in [11] for solving robust optimization ($\mathcal{RO}$) problem [4]. Specifically, we introduce the stochastic variant of deterministic saddle-point dynamical system (SDS) referred to as SSDS for robust learning. The stochastic variant is useful in stochastic gradient descent (SGD) setting, typical for training of DNN. The min-max optimization problem that arises in adversarial training of DNN is formulated as $\mathcal{RO}$ problem. The objective of the proposed SSDS algorithm is to converge to the min-max saddle point.

Unlike existing approaches, our proposed algorithm does not decouple the minimization and maximization problems involved in robust optimization. Instead, it attempts to solve both problems simultaneously by evolving both the model parameters and the adversarial perturbations through the training epochs. As we do not attempt to find the worst-case perturbations at every training epoch, we save significant computation overhead as compared to other methods such as PGD-training [18] and TRADES [32]. While there are recent efforts to mitigate the computational overhead by using random projections [29] instead of finding the worst case attacks, our approach is fundamentally different as we still try to solve the coupled robust optimization problem while reducing the computations overhead. To this end, in addition to model parameters and the adversarial perturbations, we also evolve two Lagrangian multipliers through the training epochs, one for the model parameters and the other for the adversarial perturbations. Under the assumptions of convex cost function and concave uncertainties, we analytically show that using a diminishing step-size, our SSDS algorithm converges asymptotically to the robust optimal solution. We also present extensive experimental results using CIFAR-10 and comparison with the state-of-the-art for validation.

Contributions: Specifically, our contributions are: (i) A new saddle-point dynamical systems approach to robust learning for finding a robust model and the corresponding worst-case adversarial perturbations, (ii) a new Stochastic Saddle-point Dynamical Systems (SSDS) algorithm appropriate for robust deep learning, (iii) analysis of convergence for SSDS under certain restrictive assumptions (iv) empirical results to show that the proposed approach is a computationally inexpensive way to train robust models for white- and black-box attacks as well as maintain a relatively high level of performance for clean data.

Related Work

Due to space constraints and a large amount of recent progress on adversarial machine learning, our discussion of related work is necessarily incomplete. Here, we attempt to discuss the most recent & relevant literature. We divide the section as: (1) adversarial attack/defense and (2) robust optimization.

Adversarial Attack and Defense: Initial evidence of the vulnerability of deep classifiers to imperceptible adversarial perturbations was shown by [27]. Around the same time, Biggio *et al*. [5] showed that SVMs could malfunction in security-sensitive applications and proposed a regularization term in the classifiers. In the deep learning community, Goodfellow *et al*. [13] and Kurakin *et al*. [17] proposed the Fast Gradient Sign Method (FGSM) and its iterative variants as powerful attack strategies to fool deep learning models. While these methods mainly focused on white box attacks, Papernot *et al*. [21] introduced the notion of black-box attacks where the adversary does not have complete knowledge of the learning model. Attacks can also be categorized into test-time [5] and train-time (also known as data poisoning) attacks [16], and targeted and non-targeted attacks [2]. In this paper, we only focus on test-time, non-targeted attacks.

Several defense approaches have been proposed in the literature, such as using denoising autoencoders-based Deep Contractive Networks [14], and defining a network robustness metric [3]. However, as discussed in the introduction, the most popular robust deep learning methods involve some form of adversarial training [13,18,23,28]. Defensive distillation [20] is also another method of defense which showed fascinating results. However, Carlini and Wagner [8] could break such a defense mechanism by proposing multiple adversarial loss functions. Athalye *et al*. [1] further analyzed various defense approaches and demonstrated that most existing defenses could be beaten by approximating gradients over defensively

trained models. In this paper, we only focus on defense for perception models such as deep CNN.

Robust Optimization: Authors in[23] show that adversarial training of neural networks is, in fact, robustification of the network optimization that can be exploited to increase the local stability of neural networks. Robust optimization has also been used in[9] to find an approximately optimal min-max solution that optimizes for non-convex objectives. This method is based on a reduction from robust optimization to stochastic optimization. Here an α-approximate stochastic oracle is given, and α-approximate robust optimization in a convexified solution space is obtained. Nonetheless, ideas from robust optimization (closely related to regularization in machine learning[26,30]) for solving robust learning problems has not been explored sufficiently.

Problem Formulation

We first state the robust learning problem from a robust optimization viewpoint to provide the framework for solving robust deep learning problems with the saddle-point dynamics approach.

Robust Learning as a Robust Optimization Problem

We consider a standard classification task under a data distribution $\mathcal{D}$ over the dataset $I = \{I^{(1)}, I^{(2)}, \cdots, I^{(N)}\}$, where, $I^{(i)} \in \mathbf{R}^m$ with set of labels, y. The loss function (*e.g.*, cross-entropy loss) is denoted by $L(I, y, w)$ with $w \in \mathbf{R}^n$ as the model parameters (decision variables). From a robust optimization ($\mathcal{RO}$) perspective[11], robust learning can be written as

$$\mathcal{RO} := \min_{w} \ \mathbb{E}_{(I,y)\sim\mathcal{D}} \left[\max_{u\in\mathcal{U}} \ L(I+u, y, w) \right], \tag{1}$$

where, the loss function L is also a function of additive perturbation or uncertainty u (constrained by uncertainty set $\mathcal{U}$) to the input.

Following the standard practice in machine learning, we approximate the expected loss with empirical loss for a finite number of i.i.d training samples, $I^{(i)}$ for $i \in \{1, 2, \cdots N\}$. We consider $u^{(i)}$ as the corresponding uncertainty for the data point $I^{(i)}$. Hence, $\mathcal{RO}$ problem (1) can be written as

$$\mathcal{RO} := \min_{w} \sum_{i=1}^{N} \max_{u^{(i)}\in\mathcal{U}^{(i)}} \ L(I^{(i)} + u^{(i)}, y^{(i)}, w), \tag{2}$$

The fundamental assumption in $\mathcal{RO}$ is that the uncertainty variables reside within the uncertainty sets

$$\mathcal{U}^{(i)} := \{u^{(i)} \in \mathbf{R}^m : h^{(i)}(u^{(i)}) \leq 0\}, \ i = 1, \ldots, N,$$

where the $h^{(i)}$ functions representing the uncertainty sets are typically assumed to be convex functions such as norm-bound budgets. The goal here is to obtain model parameters, w, (e.g., weights, θ and biases, b for neural networks) that work well for all possible uncertainty parameter realizations within the uncertainty sets.

We assume that the $\mathcal{RO}$ problem (1) has at least one robust feasible solution. Further, we make the following assumption for the functions in the $\mathcal{RO}$ problem (1)[11].

Assumption 1. *$L(I + u, y, w)$ is strictly convex in w and each $L(I^{(i)} + u^{(i)}, y^{(i)}, w)$ is strictly concave in $u^{(i)}$. Moreover, each $h^{(i)}(u^{(i)})$ is convex in $u^{(i)}$, and each $\mathcal{U}^{(i)}$ needs to be a compact (and convex) set for $i = 1, \ldots, N$. Moreover, norm of L and $h^{(i)}$ s and their subgradients are bounded on compact sets.*

Based on the epigraph form of an optimization problem[7], we rewrite $\mathcal{RO}$ problem (1) as

$$\mathcal{RO} := \min_{w,t} \ t \ \text{s.t} \qquad \sum_{i=1}^{N} \max_{u^{(i)}\in\mathcal{U}^{(i)}} L(I^{(i)} + u^{(i)}, y^{(i)}, w) - t \leq 0, \tag{3}$$

which is an equivalent, albeit more convenient form for our framework, where t is being added to the vector of model parameters as an auxiliary decision variable. We define the saddle and KKT point of the $\mathcal{RO}$ problem later and discuss their properties briefly.

Defining a Lagrangian multiplier $\lambda \geq 0$ and the vector of model parameters as $x := (w, t)$, the optimization problem (3) can be written as

$$\min_{x=(w,t)} \max_{\lambda\geq 0} \left\{ t + \lambda \left(\sum_{i=1}^{N} \max_{u^{(i)}\in\mathcal{U}^{(i)}} L(I^{(i)} + u^{(i)}, y^{(i)}, w) - t \right) \right\}.$$

Then the total Lagrangian for the $\mathcal{RO}$ problem can be written as

$$\mathcal{L}(x, \lambda, u, v) := t + \lambda \left(\sum_{i=1}^{N} \left(L(I^{(i)} + u^{(i)}, y^{(i)}, w) - v^{(i)} \ h^{(i)}(u^{(i)}) \right) - t \right). \tag{4}$$

where $v^{(i)}$ s are the Lagrangian multipliers for the lower level maximization problem. Derivation of the Lagrangian function along with the definition and properties of the saddle and KKT point of $\mathcal{RO}$ problem can be found in the supplementary material. Note that the proposed discrete framework for training adversarial deep learning models stems from the continuous-time dynamical system concepts and tools (e.g., Lyapunov theory) for solving robust optimization (RO)[11]. Typically, loss function in an RO framework is considered to be a function of just the model parameters, not the uncertainty. Therefore, we formulate the algorithm with Lagrangian multipliers that ensure satisfaction of the attack budget asymptotically without hard projections. This formulation also helps in analysis.

Saddle-Point Dynamical System Algorithm

Following the discussion in the previous section, we now propose the algorithms for both deterministic and stochastic saddle-point dynamical system. For simplicity, we are going to call them "SDS" for deterministic and "SSDS" for stochastic case.

Deterministic Saddle-point Dynamical System (SDS) Algorithm

Defining $x = (w, t)$, the discrete-time saddle-point dynamics method finds the saddle point of the Lagrangian function by the iterations

$$x_{k+1} = x_k - \alpha_k \Big(\partial_x t_k + \lambda_k \partial_x \textstyle\sum_{i=1}^{N} L(I^{(i)} + u_k^{(i)}, y^{(i)}, w_k) - \partial_x t_k \Big) , \tag{5}$$

$$\lambda_{k+1} = \Big[\lambda_k + \alpha_k \Big(\textstyle\sum_{i=1}^{N} (L(I^{(i)} + u_k^{(i)}, y^{(i)}, w_k) - v^{(i)}\, h^{(i)}(u^{(i)})) - t_k \Big) \Big]_+ , \tag{6}$$

$$u_{k+1}^{(i)} = u_k^{(i)} + \alpha_k \big(\partial_{u^{(i)}} L(I^{(i)} + u_k^{(i)}, y^{(i)}, w_k) - v^{(i)}\, \partial_{u^{(i)}} h^{(i)}(u^{(i)}) \big) , \tag{7}$$

$$v_{k+1}^{(i)} = [v_k^{(i)} + \alpha_k \lambda_k h^{(i)}(u_k^{(i)})]_+, \quad i = 1, \ldots, N . \tag{8}$$

Noting the $u^{(i)}$ update, we observe that the above saddle discrete dynamics for finding the robust optimal solution of $\mathcal{RO}$ problem differs from the primal-dual dynamics for deterministic problems[12]. This is because the vector field for the above algorithm is not obtained by taking the total Lagrangian's sub-gradient. Defining a set-valued mapping including subgradient functions of the Lagrangian function (4) components as

$$T(z) := \big(\partial_x \mathcal{L}(z), -\partial_\lambda \mathcal{L}(z), -\partial_{u^{(1)}} \mathcal{L}^{(1)}(x, u^{(1)}, v^{(1)}), \ldots, -\partial_{u^{(N)}} \mathcal{L}^{(N)}(x, u^{(N)}, v^{(N)}), \partial_{v^{(1)}} \mathcal{L}(z), \ldots, \partial_{v^{(N)}} \mathcal{L}(z)\big) ,$$

and denoting $\mathcal{L}^{(i)}(x, u^{(i)}, v^{(i)}) := L(I^{(i)} + u^{(i)}, y^{(i)}, w) - v^{(i)}\ h^{(i)}(u^{(i)})$ for $i = 1 \ldots, N$, the adaptive diminishing step-size in the above algorithm is defined as

$$\alpha_k = \frac{\gamma_k}{\|T(z_k)\|_2} \text{ , with } \gamma_k > 0 \text{ , } \textstyle\sum_{k=1}^{\infty} \gamma_k = \infty \text{ , } \sum_{k=1}^{\infty} \gamma_k^2 < \infty . \tag{9}$$

Following theorem is the main result for asymptotic convergence of the discrete-time saddle point algorithm with diminishing step-size. We show that algorithm (5)-(8) converges to the KKT point (equivalent to the saddle point as specified in the supplementary material) of $\mathcal{RO}$ problem.

Theorem 2. *Under Assumption 1 and the additional assumption that $\lambda^\star > 0$ where $\lambda^\star$ is the saddle point of the Lagrangian (4) for λ part, the saddle point dynamics (5)-(8) converges asymptotically to the robust optimal solution with the adaptive diminishing step-size satisfying (9).*

Remark 1. *Although the stability is not shown in this paper, we observe in practice that the dynamics without λ in v-update (8) works for both active and inactive constraints (whether $\lambda^\star$ is positive or zero) of the $\mathcal{RO}$ problem and converges to the KKT point. Therefore, we propose removing λ from v-update to use in practice for solving the $\mathcal{RO}$ problem.*

Remark 2. *Convergence results are also proved where the adaptive step-size is replaced with constant step-size, say α, in the algorithm (5)-(8). In particular, with a constant step-size, convergence rate of $o(\frac{1}{k})$ can be proved for the Lagrangian function $\mathcal{L}$ in (4) as*

$$\mathcal{L}(x^\star, \lambda^\star, u^\star, v^\star) - \frac{1}{k} \textstyle\sum_{j=1}^{k} \mathcal{L}(x_j, \lambda_j, u_j, v_j)| \leq \frac{c_1}{k} + c_0(\alpha), \tag{10}$$

where $(x^\star, \lambda^\star, u^\star, v^\star)$ specifies the saddle point, c_1 is some constant, and $\lim_{\alpha \to 0} c_0(\alpha) \to 0$.

Stochastic Saddle-point Dynamical System (SSDS) Algorithm

The deterministic saddle point algorithm presented above is modified to account for the stochastic effect inherent in the implementation of optimization algorithms (e.g., SGD) involving a large training set. To reduce the computational burden of calculating batch gradients for a large training set, typically the gradient of loss function is computed for a randomly selected data point (simple SGD) or a small batch of data points (mini-batch SGD). Following similar works in[6] for a simple SGD setting, we define $x := (t, w)$, $f(x) := t$, $g(x, u, \xi) := L(I_\xi + u^\xi, y, w) - t$, where $\xi \in \{1, \ldots, N\}$ is a random variable modeling the process for randomly selecting a data point out of N possible samples. Furthermore, we define $g(x, u) = \sum_i^N L(I^{(i)} + u^{(i)}, y^{(i)}, w) - t$. With these notations, the stochastic version of the deterministic saddle point algorithm is written as follows:

$$x_{k+1} = x_k - \alpha_k (\partial_x f(x_k) + \lambda_k \partial_x g(x_k, u_k, \xi_k)) , \tag{11}$$

$$\lambda_{k+1} = [\lambda_k + \alpha_k (g(x_k, u_k, \xi_k) - \textstyle\sum_{i=1}^{N} v_k^{(i)} h^{(i)}(u_k^{(i)}))]_+ , \tag{12}$$

$$u_{k+1}^{(i)} = u_k^{(i)} + \alpha_k (\partial_{u^{(i)}} g(x_k, u_k) - v_k^{(i)} \partial_{u^{(i)}} h^{(i)}(u_k^{(i)})) , \tag{13}$$

$$v_{k+1}^{(i)} = [v_k^{(i)} + \alpha_k \lambda_k h^{(i)}(u_k^{(i)})]_+ \quad i = 1, \ldots, N. \tag{14}$$

where, ξ_k is assumed to be independent identically distributed random process and α_k is the adaptive step-size in (9), and following assumptions are made on f, g and $h^{(i)}$ s. Note that the weights x is updated using the loss function information at randomly selected image u^ξ, however the uncertain variable u^i corresponding to all the images are updated.

Assumption 3. *We assume that $f(x)$ is convex in x and each $h^{(i)}(u^{(i)})$ is convex in $u^{(i)}$. Moreover, $g(x, u, \xi)$ is*

convex in x and is strictly concave in u for any fixed value of ξ.

Remark 3. *Clearly, the assumptions of strict convexity of f and concavity of g are not satisfied in the DNN setting. The strict convexity and strict concavity assumptions could be relaxed with weaker convergence results than the one reported below. This is the topic of current investigation.*

Theorem 4. *Let Assumption 3 be satisfied and $\lambda^\star > 0$, then, following is true for the SSDS algorithm with adaptive step-size α_k satisfying (9).*

$$\lim_{k\to\infty} \mathbb{E}_{\xi_0^k}[x_k] = x^\star, \lim_{k\to\infty} \mathbb{E}_{\xi_0^k}[u_k] = u^\star \text{ , where } \quad \xi_0^k = \{\xi_0, \ldots, \xi_k\}. \tag{15}$$

Algorithm 1 Mini-batch SSDS algorithm

Input: ε, lr, p, C_1, C_2
Initialization: λ_0, α_0,w_0, t_0,u_0, v_0
for $k \in \{1, ..., K\}$ **do**
  distribute mini-batches as $m = \{m_0, m_1, .., m_n\}$
  $w_k^{(m_0)} = w_k$
  $\lambda_k^{(m_0)} = \lambda_k$
  for $m_j \in \{m_0, m_1, ..., m_n\}$ **do**
    $\partial_{w_k} = \partial_{w_k} \sum_{j\in m_j} L(I^{(j)} + u_k^{(j)}, y^{(j)}, w_k^{(j)})$
    $w_k^{(m_{(j+1)})} = w_k^{(m_{(j)})} - lr\ \lambda_k(\partial_{w_k})$
  end for
  $t_{k+1} = t_k + \alpha_k(\lambda_k - 1)$
  $v_{k+1}^{(m_j)} = v_k^{(m_j)} + \alpha_k(\|u_k^{(m_j)}\|_\infty - \varepsilon)$
  $\partial_{u_k} = \partial_{u_k} L(I^{(m_j)} + u_k^{(m_j)}, y^{(m_j)}, w_k^{(m_j)})$
  $u_{k+1}^{(m_j)} = u_k^{(m_j)} + \alpha_k(\partial_{u_k} - C_1\, v_k^{(m_j)}\, \text{sign}(u_k^{(m_j)}))$
  for $j \in m_j$ **do**
    $B^{(j)} = (\|u_k^{(j)}\|_\infty - \varepsilon)$
    $U^{(j)} = \big(L(I^{(j)} + u_k^{(j)}, y^{(j)}, w_k^{(j)}) - v_k^{(j)}\big) B^{(j)}$
  end for
  $U = \sum_{j\in m_j} U^{(j)}$
  $\lambda_{k+1}^{(m_j)} = \lambda_k^{(m_j)} + C_2\, \alpha_k \Big(U - t_k\Big)$
  $w_k = w_k^{(m_n)}$
  $\lambda_k = \lambda_{k+1}^{(m_n)}$
  $\alpha_{k+1} = \alpha_k e^{-kp}$
end for

Mini-batch Implementation of SSDS Algorithm

In an attempt to use the proposed approach for robust training of DNN, we propose a mini-batch stochastic version of the SSDS in algorithm 1 to achieve a more stable convergence. As stated above, the SSDS algorithm involves the decision variable $x := (t, w)$, where ws are the parameters of DNN. For simplicity of implementation, we first separate the update rule for x, described in (11). In other words, we split the updates of w and t that also enables us to use standard learning rates (denoted by lr) for the w updates. For the updates of t and other SSDS variables, such as λ, u and v, we use a diminishing step-size α_k. However, we refrain from applying the diminishing step-size described in (9) due to the sheer complexity involved in taking the norm of the parameters for a large-scale neural network. Instead, we use an exponentially decaying diminishing step-size, $\alpha_{k+1} = \alpha_k e^{-kp}$, where p is the decay rate for exponentially diminishing stepsize and k is the epoch number. Note that the updates of u and λ can succumb to scaling issues depending of the values of the gradients and variable v_k. Therefore, we add two scaling factors C_1 and C_2 in the update rules of u and λ to bring different terms of the update laws to the same scale. Given a data set and a model architecture, appropriate values can be found with a few trial and error steps.Due to the separation of the updates, another small departure in our implementation from the prescribed algorithm is - while w updates are performed for every mini-batch (m_j refers to the j^{th} mini-batch of k^{th} epoch) , the other updates are performed once every epoch. Also, in the original formulation, we continuously update u corresponding to all the images while the w is updated using the gradient information of the loss function evaluated at randomly selected images. In the algorithm implementation however, u is also updated only corresponding to the randomly selected images based on which the network weights are updated. This helps in reducing computation for large training sets.

Based on the above setup, the mini-batch SSDS algorithm is presented in Algorithm 1.

We can also craft attacks based on the SSDS algorithm. To do that, we run iterative updates of the perturbations u given a test sample along with its corresponding Lagrangian multiplier v, keeping the model ($w^\star$) and λ^* fixed. The attack algorithm is discussed in the supplementary material.

Experimental Results

Most robust learning papers consider only CIFAR-10 and MNIST, where performances on MNIST data sets are usually difficult to differentiate. We see the same trend for MNIST as seen with CIFAR-10. Therefore, We validate the proposed SSDS algorithm empirically on the CIFAR-10 dataset Specifically, we demonstrate the convergence characteristics of the algorithm along with performance comparison with benchmark techniques. We use the Resnet 50 [15] and VGG19[24] model architectures and the software implementation is done in Pytorch[22]. The codes will be made available upon acceptance of the paper. The key hyper-parameters of the proposed algorithm are chosen as: $lr = 0.001$, $\varepsilon = 0.03$, $p = 0.001$, $C_1 = C_2 = 0.01$. Through repeated trial and error, we find the following suitable initializations: $\lambda_0 = 4$, $\alpha_0 = 2$, $t_0 = 0$ and $u_0 = 0$, $v_0 = 1$, for all input images.

SSDS convergence characterisitcs

We begin this discussion with general training accuracy and loss plots shown in Fig. 1 for mini-batch SSDS algorithm for the CIFAR-10 dataset using VGG19 model architecture. The ℓ_∞-norm attack budget was chosen to be 0.03 or 3% of the maximum pixel intensity (comparable with previous works in this area [18,33]). We also observe the accuracy values for the clean test set during the training process. Additionally, we plot the histogram of ℓ_∞-norms of final perturbations added to the training images for a few epochs during the training process (see Fig. 1c). This is to verify the theoretical claim that the final perturbations for the training images should converge at or below the budget. From the empirical results shown in Fig. 1c, we make the observation that although perturbations for some images spill over the threshold value (0.03) during the course of the training process, most of the perturbations converge within the bound eventually (interestingly, a large number of the perturbations settle below the threshold).

Next, we focus on a specific training sample to understand how the dynamics for different variables in the algorithm evolve during the course of the training process. A randomly chosen training image is shown in Fig. 2a along with the corresponding final perturbation generated by the algorithm and corrupted adversarial version of the image. We observe the dynamics of v over the training epochs. In this experiment, v was initialized at 1 (for all of the images in the training set), and it converges to 0 after around epoch 180. Similarly, as shown in Fig. 2c, ℓ_∞-norm of the perturbation generated for the chosen image converges to 0.026 which is below the specified budget. However, the actual perturbations for individual pixels still continue to evolve even after the ℓ_∞-norm for the entire perturbation matrix settles down. To monitor the perturbations for the individual pixels, we plot ℓ_2 norm of the difference between the perturbations for two consecutive epochs. We see that this metric finally converges to 0 around epoch 300. At this point, the overall training process also converges except for small changes due to the stochastic nature of the training algorithm.

Model evaluation

We begin with performance comparison of SSDS-p with PGD-training (trained with 20 PGD steps and 10 random starts [18,33]) and FGSM-training [17] on various white-box attacks, such as FGSM, 20 step PGD and SSDS-p attacks (see Table 1). We also keep the record for naturally trained models as a baseline. We present our results with two different model architectures: ResNet and VGG.

Table 1 *Defense method comparison under white-box attacks (Resnet50 model architecture)*

Attack	Accuracy
Clean	84.75%
FGSM	40.48%
PGD	12.80%
SSDS-p	17.51%

(a) Natural training

Attack	Accuracy
Clean	52.96 %
FGSM	70.12 %
PGD	9.61%
SSDS-p	10.83 %

(b) FGSM training

Attack	Accuracy
Clean	45.52%
FGSM	61.81%
PGD	39.49%
SSDS-p	43.06%

(c) PGD training

Attack	Accuracy
Clean	57.12%
FGSM	**71.23%**
PGD	**46.11%**
SSDS-p	**55.55%**

(d) SSDS-p training

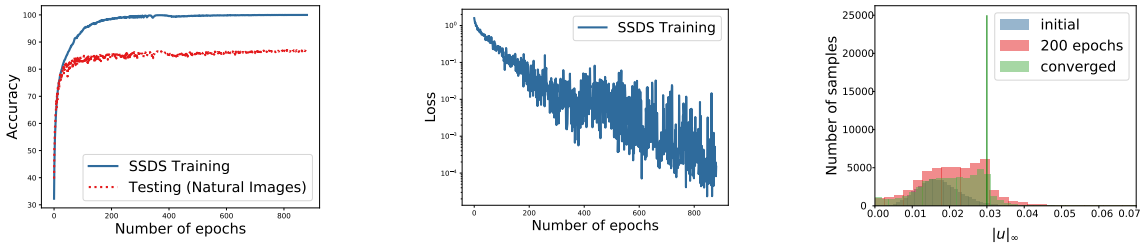
Resnet Model: Table 1 summarizes the results for ResNet50 models under white-box attacks. We observe that SSDS-p training performs significantly better than Natural training, FGSM-training and 20 step PGD training for all attacks. While outperforming FGSM and PGD training for clean data, SSDS-p still has considerably lower performance in comparison with the natural model for clean images. However, this is a well-known observation for standard ResNet models, which is typically addressed by using a wider model [18].

Table 2 *Black box table-model: Resnet50 without pretraining, Optimizer=SGD, attack= 7-step PGD, $\varepsilon = 3\%$*

Target	Source	Accuracy
20-step PGD	Naturally-trained	43.40%
SSDS-p	Naturally-trained	**51.11%**
20-step PGD	SSDS-p	42.18%
SSDS-p	20-step PGD	**45.31%**

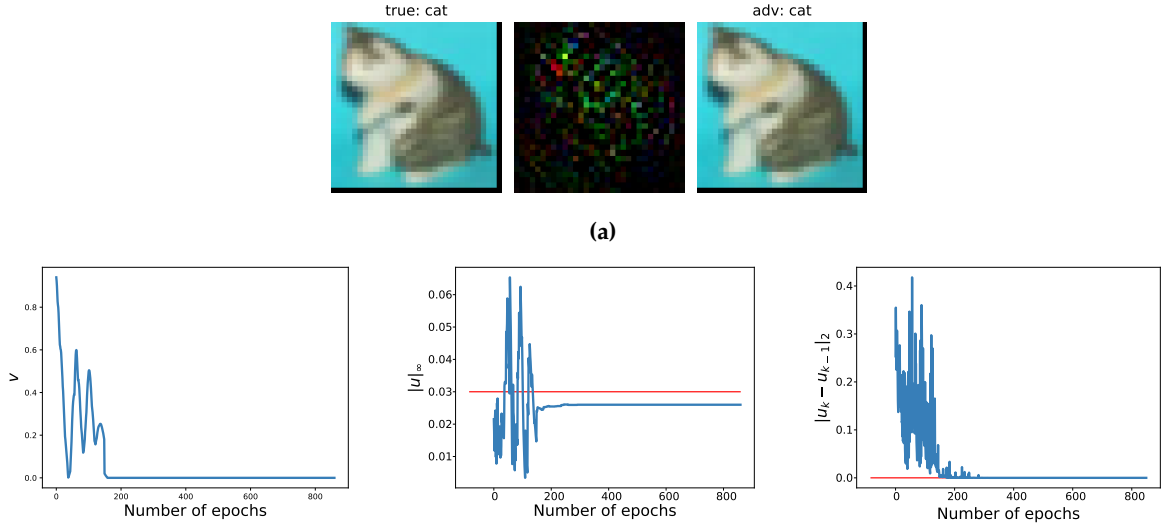
Next, we evaluate our defense method for black box attacks. We use our (SSDS-p) Resnet50 model for defending against 7 step PGD attacks generated using a naturally trained model. From Table 2, we observe that our accuracy is significantly better compared that of a 20 step-PGD model with the same architecture. Then we perform a head-to-head comparison where we try to defend against 7 step PGD attacks generated using a 20 step-PGD model with SSDS-p training and vice-versa. The SSDS-p training performs better in this case as well (See Table 2).

We then extend our experiments to the running time comparison between SSDS and PGD. Fig. 3 shows that although PGD can acheive comparable acuracies with SSDS-p, the training time per epoch for 20 step PGD-training algorithm is approximately 16 times greater than that of the SSDS-p training algorithm. On the other hand, SSDS-p training takes around the same time as a 1 step PGD training. Note, we do not compare our method with TRADES [32] as



(a) **(b)** **(c)**

Figure 1 *Results on CIFAR-10 dataset using VGG19 model: (a) Accuracy (b) Loss value* (log *scale) (c)* $\|u\|_\infty$ *histogram*



(a)

(b) **(c)** **(d)**

Figure 2 *Results on CIFAR-10 dataset using VGG19 model: (a) Example of SSDS training image(left), with its corruption(center) and the corrupted image(right) (b) values of v for the above image (c) Evolution of* $\|u\|_\infty$ *for the above image (d) Evolution of* $\|u_k - u_{k-1}\|_2$ *for the above image*

that algorithm is extremely similar to PGD training (except for a new loss function for improved adversarial accuracy) from a computational perspective. Therefore, our approach still remains fundamentally different and computationally significantly more efficient.

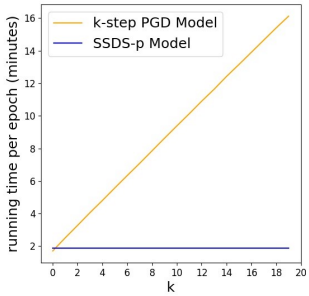


Figure 3 *Training time per epoch for PGD vs. SSDS-p (Resnet50 model architecture)*

VGG model: We now use a VGG19 model to evaluate our algorithm. While VGG19 training is computationally less expensive compared to ResNet50, SSDS-p training still remains significantly more efficient compared to multi-step PGD training (see Fig 6 in Supplementary material). We observe that SSDS-p training performs significantly better than FGSM-training for all attacks (see Table 3). SSDS-p significantly outperforms 20 step PGD-training on clean and FGSM corrupted images as well. Interestingly, SSDS-p performance on clean data is very close to that of natural training. However, PGD-training works better for PGD corrupted images. On the other hand, SSDS-p training performs better in classifying the SSDS-p corrupted images compared to PGD-training. We also used our defense model to defend against test cases generated by Madry *et al.* [18] models which are discussed in the Supplementary material.

To further explore the trade-off between computa-

Table 3 *Defense method comparison under white-box attacks (VGG19 model architecture)*

Attack	Accuracy
Clean	86.10%
FGSM	53.70%
PGD	24.64%
SSDS-p	27.43%

(a) Natural training

Attack	Accuracy
Clean	86.42 %
FGSM	54.92 %
PGD	23.10%
SSDS-p	30.97 %

(b) FGSM training

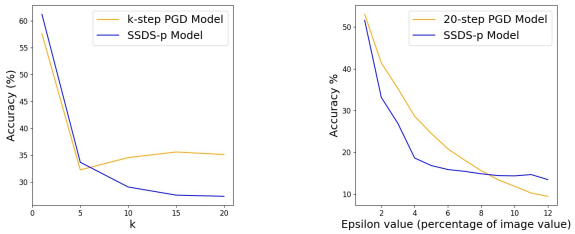
Attack	Accuracy
Clean	69.11%
FGSM	45.42%
PGD	**35.44%**
SSDS-p	41.98%

(c) PGD training

Attack	Accuracy
Clean	85.52%
FGSM	**60.95** %
PGD	27.61%
SSDS-p	**53.54** %

(d) SSDS-p training

tional cost and accuracy, we extend our experiments to different computation and attack budgets. We used different steps of PGD models to compare our model with a less computationally expensive one. In order to compare the performance of SSDS-p model with k step PGD model for various k values, we train different k step PGD models and test each of them with their corresponding attack protocols, e.g., we test a 5 step PGD model with 5 step PGD attack and so on. As shown in Fig. 4a, we see that the SSDS-p model outperforms k step PGD models until $k = 5$, while being significantly better in terms of computation overhead. We also note that apart from training time per epoch, SSDS-p usually converges in much less number of epochs as compared to PGD training. Therefore, the total training time becomes even shorter for SSDS-p. Finally, we consider the performance of our model trained with ℓ_∞ budget of 0.03 under different attack budgets. In[18], authors observed that as the attack budget (ε) increases, the 20 step PGD model loses accuracy almost exponentially for the CIFAR-10 data set. In contrast, SSDS-p model accuracy saturates after 4%, eventually outperforming PGD-training around $\varepsilon = 8\%$. (see Fig. 4b)



(a) **(b)**

Figure 4 *SSDS-p and PGD comparison (VGG model architecture): (a) Performance comparison for PGD-k vs. SSDS, (b) Testing accuracy of 20-step PGD model vs. SSDS-p model with 3% budget on 20-step PGD corrupted images with different budgets*

Conclusion

In this paper, we propose a new saddle-point dynamical systems approach to solve the robust learning problem. Under certain restrictive assumptions, we present a detailed convergence analysis for the stochastic version of our algorithm. Empirically, we show that the proposed scheme is a computationally inexpensive method that maintains a high level of performance for clean and corrupted input data, both for white-box and black-box attacks We believe that this can be attributed to the fact that the adversarial training in SSDS also acts as a form of regularization. This notion is based on the equivalence between the robust optimization problem and many regularization problems[26]. Finally, we note that this is an early attempt to adopting a dynamical systems approach to robust learning. Therefore, future research will focus on relaxing some of the restrictive assumptions in the analysis for the loss function and uncertainties. Similarly, we will focus on developing the SSDS algorithm further to better handle the highly non-convex nature of deep network loss functions, leading to a more competitive performance with computationally intensive adversarial training processes.

References

[1] Athalye, A.; Carlini, N.; and Wagner, D. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*.

[2] Barreno, M.; Nelson, B.; Joseph, A. D.; and Tygar, J. D. 2010. The security of machine learning. *Machine Learning* 81(2):121–148.

[3] Bastani, O.; Ioannou, Y.; Lampropoulos, L.; Vytiniotis, D.; Nori, A.; and Criminisi, A. 2016. Measuring neural net robustness with constraints. In Lee, D. D.; Sugiyama, M.; Luxburg, U. V.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc. 2613–2621.

[4] Ben-Tal, A.; Hertog, D. D.; and Vial, J. P. 2015. Deriving robust counterparts of nonlinear uncertain inequalities. *Mathematical Programming, Vol. 149, No. 1* 265–299.

[5] Biggio, B.; Corona, I.; Maiorca, D.; Nelson, B.; Šrndić, N.; Laskov, P.; Giacinto, G.; and Roli, F. 2013. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, 387–402. Springer.

[6] Bottou, L.; Curtis, F. E.; and Nocedal, J. 2018. Optimization methods for large-scale machine learning. *Siam Review* 60(2):223–311.

[7] Boyd, S., and Vandenberghe, L. 2004. *Convex Optimization*. New York: Cambridge Univ. Press.

[8] Carlini, N., and Wagner, D. 2017. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, 39–57. IEEE.

[9] Chen, R. S.; Lucier, B.; Singer, Y.; and Syrgkanis, V. 2017. Robust optimization for non-convex objectives. In *NIPS*.

[10] Danskin, J. M. 1966. The theory of max-min, with applications. *SIAM Journal on Applied Mathematics* 14(4):641–664.

[11] Ebrahimi, K.; Elia, N.; and Vaidya, U. 2019. A continuous time dynamical system approach for solving robust optimization. In *European Control Conference, Naples, Italy*.

[12] Feijer, D., and Paganini, F. 2010. Stability of primal-dual gradient dynamics and applications to network optimization. *Automatica 46*.

[13] Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

[14] Gu, S., and Rigazio, L. 2014. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*.

[15] He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep residual learning for image recognition. *CoRR* abs/1512.03385.

[16] Khalid, F.; Hanif, M. A.; Rehman, S.; and Shafique, M. 2018. Security for machine learning-based systems: Attacks and challenges during training and inference. In *2018 International Conference on Frontiers of Information Technology (FIT)*, 327–332. IEEE.

[17] Kurakin, A.; Goodfellow, I.; and Bengio, S. 2016. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*.

[18] Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

[19] Moosavi-Dezfooli, S.-M.; Fawzi, A.; and Frossard, P. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2574–2582.

[20] Papernot, N.; McDaniel, P.; Wu, X.; Jha, S.; and Swami, A. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, 582–597. IEEE.

[21] Papernot, N.; McDaniel, P.; Goodfellow, I.; Jha, S.; Celik, Z. B.; and Swami, A. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, 506–519. ACM.

[22] Paszke, A.; Gross, S.; Chintala, S.; and Chanan, G. 2017. Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration. *PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration* 6.

[23] Shaham, U.; Yamada, Y.; and Negahban, S. 2018. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing* 307:195–204.

[24] Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[25] Sitawarin, C.; Bhagoji, A. N.; Mosenia, A.; Chiang, M.; and Mittal, P. 2018. Darts: Deceiving autonomous cars with toxic signs. *arXiv preprint arXiv:1802.06430*.

[26] Sra, S.; Nowozin, S.; and Wright, S. J. 2011. *Optimization for Machine Learning*. MIT Press.

[27] Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

[28] Tramèr, F.; Kurakin, A.; Papernot, N.; Goodfellow, I.; Boneh, D.; and McDaniel, P. 2017. Ensemble Adversarial Training: Attacks and Defenses. *arXiv e-prints* arXiv:1705.07204.

[29] Wong, E.; Schmidt, F.; Metzen, J. H.; and Kolter, J. Z. 2018. Scaling provable adversarial defenses. In *Advances in Neural Information Processing Systems*, 8400–8409.

[30] Xu, H.; Caramanis, C.; and Mannor, S. 2009. Robust regression and lasso. *In Advances in Neural Information Processing Systems, pages 18011808*.

[31] Zagoruyko, S., and Komodakis, N. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146*.

[32] Zhang, H.; Yu, Y.; Jiao, J.; Xing, E. P.; Ghaoui, L. E.; and Jordan, M. I. 2019. Theoretically principled trade-off between robustness and accuracy. *CoRR* abs/1901.08573.

[33] Zheng, T.; Chen, C.; and Ren, K. 2018. Distributionally adversarial attack. *CoRR* abs/1808.05537.

Supplementary Material

Additional proofs and experimental results will be discussed here; also more information about the material in the main body of the paper will be added here.

Additional Theorems and Derivations

We consider the following definitions in this section

$$x := (t, w),\ f(x) := t,\ g(x,u) := L(I+u, y, w) - t\ . \tag{16}$$

Lagrangian function derivation

First, we derive the Lagrangian function in (4). The lower level optimization problem in the $\mathcal{RO}$ problem (1) can be written as an optimization problem parametrized by x such that

$$\theta(x) := \textstyle\sum_{i=1}^{N} \max_{u^{(i)} \in \mathcal{U}^{(i)}} L(I^{(i)} + u^{(i)}, y^{(i)}, w) - t\ . \tag{17}$$

Denote $v^{(i)}$ s as Lagrangian multipliers for the lower level maximization problem and define $v := [v_1, \ldots, v_N]^\top$. The role of each $v^{(i)}$ multiplier is to satisfy the uncertainty set constraint associated with the perturbation $u^{(i)}$. Based on the Lagrangian theory, one can equivalently write

$$\begin{aligned}\theta(x) = \textstyle\sum_{i=1}^{N} \max_{u^{(i)}} \min_{v^{(i)} \geq 0} \big(L(I^{(i)} + u^{(i)}, y^{(i)}, w) \\ - v^{(i)}\ h^{(i)}(u^{(i)})\big) - t\ .\end{aligned} \tag{18}$$

Hence, $\mathcal{RO}$ problem can be written as

$$\begin{aligned}\mathcal{RO} = \min_{x=(w,t)} \max_{\lambda \geq 0} \Big\{\ & t + \\ & \lambda \Big(\textstyle\sum_{i=1}^{N} \max_{u^{(i)}} \min_{v^{(i)} \geq 0} \big(L(I^{(i)} + u^{(i)}, y^{(i)}, w) - \\ & v^{(i)}\ h^{(i)}(u^{(i)})\big) - t \Big) \Big\} \\ = \min_{x=(w,t)} \max_{\lambda \geq 0} \max_{u^{(i)}} \min_{v^{(i)} \geq 0} \Big\{\ & t + \\ & \lambda \Big(\textstyle\sum_{i=1}^{N} \big(L(I^{(i)} + u^{(i)}, y^{(i)}, w) - v^{(i)}\ h^{(i)}(u^{(i)})\big) - t \Big) \Big\}\ .\end{aligned} \tag{19}$$

Therefore, one can derive the total Lagrangian as in (4).

Saddle and KKT point of the $\mathcal{RO}$ problem

The following theorem can be stated for the saddle point of the optimization problem (1).

Theorem 5. *Consider the Lagrangian function as defined in (4). Under Assumption 1, following statements are true for the optimization problem (1)*

$$\mu = \min_{x} \max_{\lambda \geq 0} \max_{u^{(i)}, \forall i} \min_{v^{(i)} \geq 0, \forall i} \mathcal{L} = \min_{x} \min_{v_{i \geq 0}, \forall i} \max_{\lambda \geq 0} \max_{u^{(i)}, \forall i} \mathcal{L}\ ,$$
$$\mu = \min_{x, v^{(i)}, \forall i} \max_{\lambda \geq 0, u^{(i)}, \forall i} \mathcal{L} = \max_{\lambda \geq 0, u^{(i)}, \forall i} \min_{x, v^{(i)}, \forall i} \mathcal{L}\ ,$$

where $x \in \mathbb{R}^n, \lambda \geq 0, u^{(i)} \in \mathbb{R}^m$, and $v^{(i)} \geq 0$ for $i = 1, \ldots, N$. Hence, the Lagrangian function (4) has a saddle point.

Proof. For the ease of notations, consider the $\mathcal{RO}$ problem with single constraint and single uncertainty set as

$$\mu := \min_{x}\ f(x)\ \text{ s.t. } \max_{h(u) \leq 0}\ g(x,u) \leq 0\ . \tag{20}$$

The general case (1) with multiple uncertainty sets can be proved along similar lines. The Lagrangian for upper level problem in (20) is

$$f(x) + \lambda \left(\max_{h(u) \leq 0} g(x,u) \right)\ .$$

We can write the total Lagrangian for (20) as

$$\mathcal{L}(x, v, \lambda, u) = f(x) + \lambda \left(g(x,u) - v h(u) \right)\ .$$

Hence, we can write

$$\begin{aligned}\mu &= \min_{x} \max_{\lambda \geq 0} \max_{u} \min_{v \geq 0} \mathcal{L}(x, v, \lambda, u) \\ &= \min_{x} \max_{\lambda \geq 0} \max_{u} \min_{v \geq 0} \big(f(x) + \lambda (g(x,u) - v h(u)) \big) \\ &= \min_{x} \big(f(x) + \max_{\lambda \geq 0} \max_{u} \min_{v \geq 0} \lambda (g(x,u) - v h(u)) \big)\ .\end{aligned}$$

We now show that

$$\mu = \min_{x} \min_{v \geq 0} \max_{\lambda \geq 0} \max_{u} \mathcal{L}(x, v, \lambda, u)\ ,$$

noting the switch in the sequence of min-max. It is sufficient to show that for any x,

$$\gamma := \max_{\lambda \geq 0} \max_{u} \min_{v \geq 0} \lambda (g(x,u) - v h(u)) = \min_{v \geq 0} \max_{\lambda \geq 0} \max_{u} \lambda (g(x,u) - v h(u))\ .$$

Let

$$\mathcal{G}(x) := \max_{h(u)\le 0} g(x,u) = \max_{u} \min_{v\ge 0} \left(g(x,u) - vh(u)\right) . \tag{21}$$

So,

$$\gamma = \max_{\lambda\ge 0} \mathcal{G}(x) = \begin{cases} 0 & \mathcal{G}(x) \le 0 \\ \infty & \mathcal{G}(x) > 0 \end{cases} . \tag{22}$$

From strong duality for the parametric optimization problem (21), we have

$$\mathcal{G}(x) = \min_{v\ge 0} \max_{u} (g(x,u) - vh(u)) .$$

Now, consider the second part in (9), that is

$$\min_{v\ge 0} \max_{\lambda\ge 0} \max_{u} \lambda(g(x,u) - vh(u)) .$$

Starting from the first $\max_u$ at right, we get

$$\max_{u} \lambda(g(x,u) - vh(u)) = \begin{cases} 0 & \lambda = 0 \\ \max_{u} \lambda(g(x,u) - vh(u)) & \lambda > 0,\ v \ge 0 \end{cases} \tag{23}$$

Then, consider $\max_{\lambda\ge 0}$ as

$$\max_{\lambda\ge 0} \begin{cases} 0 & \lambda = 0 \\ \max_{u} \lambda(g(x,u) - vh(u)) & \lambda > 0,\ v \ge 0 \end{cases} = \begin{cases} \infty & v \ge 0,\ \max_{u} (g(x,u) - vh(u)) > 0 \\ 0 & v \ge 0,\ \max_{u} (g(x,u) - vh(u)) = 0 \\ 0 & v \ge 0,\ \max_{u} (g(x,u) - vh(u)) < 0 \end{cases} .$$

Lastly, consider $\min_{v\ge 0}$ as

$$\min_{v\ge 0} \begin{cases} \infty & v \ge 0,\ \max_{u} (g(x,u) - vh(u)) > 0 \\ 0 & v \ge 0,\ \max_{u} (g(x,u) - vh(u)) = 0 \\ 0 & v \ge 0,\ \max_{u} (g(x,u) - vh(u)) < 0 \end{cases} = \begin{cases} \infty & \min_{v\ge 0} \max_{u} (g(x,u) - vh(u)) > 0 \\ 0 & \min_{v\ge 0} \max_{u} (g(x,u) - vh(u) \le 0 \end{cases} ,$$

which is equal to γ as claimed in (22). Since minimizations (maximizations) can always be combined, the above result shows that

$$\mu = \min_{x,v\ge 0} \max_{\lambda\ge 0,u} \mathcal{L}(x,v,\lambda,u) .$$

Note that $\mathcal{L}$ is (jointly) convex in (x,v), but it is not (jointly) concave in (λ,u); although it is concave in each of these variables. We next show that notwithstanding this issue, the optimal solution to μ is a saddle point. Specifically, we show that

$$\mu = \min_{x,v\ge 0} \max_{\lambda\ge 0,u} \mathcal{L}(x,v,\lambda,u) = \max_{\lambda\ge 0,u} \min_{x,v\ge 0} \mathcal{L}(x,v,\lambda,u) . \tag{24}$$

To show this, note that strong duality in the upper level parametric optimization problem in (20) implies

$$\mu = \min_{x} \max_{\lambda\ge 0} \left(f(x) + \lambda\mathcal{G}(x)\right) = \max_{\lambda\ge 0} \min_{x} \left(f(x) + \lambda\ \mathcal{G}(x)\right)$$
$$= \max_{\lambda\ge 0} \min_{x} \max_{u} \min_{v\ge 0} \left(f(x) + \lambda(g(x,u) - vh(u))\right) ,$$

where the last equality comes from the definition of $\mathcal{G}(x)$ in (21). To obtain the result in (24), we need to show that for any $\lambda \ge 0$,

$$\eta = \min_{x} \max_{u} \min_{v\ge 0} \left(f(x) + \lambda(g(x,u) - vh(u))\right)$$
$$= \max_{u} \min_{x} \min_{v\ge 0} \left(f(x) + \lambda(g(x,u) - vh(u))\right) .$$

Note that

$$\min_{v\ge 0} \left(-v\lambda h(u)\right) = \begin{cases} 0 & \lambda h(u) \le 0 \\ -\infty & \lambda h(u) > 0 \end{cases} .$$

So, we have

$$\max_{u} \left(f(x) + \lambda g(x,u) + \begin{cases} 0 & \lambda h(u) \le 0 \\ -\infty & \lambda h(u) > 0 \end{cases}\right)$$
$$= f(x) + \max_{\lambda h(u)\le 0} \lambda g(x,u) .$$

Thus,

$$\eta = \min_{x} \max_{\lambda h(u)\le 0} \left(f(x) + \lambda g(x,u)\right) .$$

Since $g(x,u)$ is convex in x and concave in u as for Assumption 1, so $f(x) + \lambda g(x,u)$ has the same properties for $\lambda \ge 0$. It follows that the result does not change if we swap the order of the optimizations. Hence,

$$\eta = \min_{x} \max_{\lambda h(u)\le 0} \left(f(x) + \lambda g(x,u)\right)$$
$$= \max_{\lambda h(u)\le 0} \min_{x} \left(f(x) + \lambda g(x,u)\right)$$
$$= \max_{u} \min_{x} \min_{v\ge 0} (f(x) + \lambda g(x,u) - vh(u)) ,$$

which completes the proof of Theorem 5. □

Let $z^\star = (x^\star, \lambda^\star, u^\star, v^\star)$ be the saddle point for the Lagrangian (4). Using the result of Theorem 5, it follows that $z^\star$ enjoys the saddle point property, namely

$$\mathcal{L}(x^\star, \lambda, u, v^\star) \leq \mathcal{L}(x^\star, \lambda^\star, u^\star, v^\star) \leq \mathcal{L}(x, \lambda^\star, u^\star, v) \,. \tag{25}$$

From the above discussion on the development of Lagrangian function $\mathcal{L}$, it follows that $\mathcal{RO}$ problem can be viewed as two connected optimization problems. The lower level optimization problem (17) parameterized by x involving maximization over uncertain variables $u^{(i)}$ s and the upper level optimization problem involving minimization over the decision variable x. This insight can be used to define the Karush-Kuhn-Tucker (KKT) conditions for the $\mathcal{RO}$ problem as follows.

Definition 1. *Recalling that $x = (w, t)$, the KKT point $(x^\star, \lambda^\star, u^\star, v^\star)$ for the $\mathcal{RO}$ problem (1) can be defined as follows*

$$\begin{aligned}
\partial_x \mathcal{L}(x^\star, \lambda^\star, u^\star, v^\star) &= 0 \,, \\
\partial_{u^{(i)}} \mathcal{L}^{(i)}(x^\star, u^{(i)\star}, v^{(i)\star}) &= 0 \,, \qquad (26) \\
\lambda^\star &\geq 0, \\
\lambda^\star \big(L(I^{(i)} + u^{(i)\star}, y^{(i)}, w^\star) - & \\
t^\star - v^{(i)\star}\, h^{(i)}(u^{(i)\star}) \big) &= 0 \,, \qquad (27) \\
v^{(i)\star} &\geq 0, \\
v^{(i)\star}\, h^{(i)}(u^{(i)\star}) &= 0 \,, \qquad (28) \\
L(I^{(i)} + u^{(i)\star}, y^{(i)}, w^\star) - & \\
t^\star - v^{(i)\star}\, h^{(i)}(u^{(i)\star}) &\leq 0 \,, \\
h^{(i)}(u^{(i)\star}) &\leq 0 \,, \qquad (29)
\end{aligned}$$

for $i = 1, \ldots, N$, where $\partial_x f$ is the notation for the gradient of f w.r.t. x, $\mathcal{L}$ defined in (4), and $\mathcal{L}^{(i)}(x, u^{(i)}, v^{(i)}) := L(I^{(i)} + u^{(i)}, y^{(i)}, w) - v^{(i)}\, h^{(i)}(u^{(i)})$ for $i = 1 \ldots, N$.

We now propose the following fundamental theorem on establishing the connection between the KKT and saddle point of the $\mathcal{RO}$ problem.

Theorem 6. *The KKT point $(x^\star, \lambda^\star, u^\star, v^\star)$ satisfying conditions (26)-(29) also satisfies saddle point inequalities in (25) and vice versa.*

Proof. Considering the definitions in (16), we first show that the KKT point satisfies the saddle point property. Note that

$$\begin{aligned}
&\mathcal{L}(x^\star, \lambda^\star, u^\star, v^\star) - \mathcal{L}(x^\star, \lambda, u, v^\star) \\
&= \lambda^\star (g(x^\star, u^\star) - \textstyle\sum_{i=1}^N v^{(i)\star} h^{(i)}(u^{(i)\star})) - \\
&\quad \lambda (g(x^\star, u) - \textstyle\sum_{i=1}^N v^{(i)\star} h^{(i)}(u^{(i)})) \\
&= (\lambda^\star + \lambda - \lambda)(g(x^\star, u^\star) - \textstyle\sum_{i=1}^N v^{(i)\star} h^{(i)}(u^{(i)\star})) - \\
&\quad \lambda (g(x^\star, u) - \textstyle\sum_{i=1}^N v^{(i)\star} h^{(i)}(u^{(i)})) \\
&= \lambda (g(x^\star, u^\star) - \textstyle\sum_{i=1}^N v^{(i)\star} h^{(i)}(u^{(i)\star})) - \\
&\quad \lambda (g(x^\star, u) - \textstyle\sum_{i=1}^N v^{(i)\star} h^{(i)}(u^{(i)})) + \\
&\quad (\lambda^\star - \lambda)(g(x^\star, u^\star) - \textstyle\sum_{i=1}^N v^{(i)\star} h^{(i)}(u^{(i)\star})) \,.
\end{aligned}$$

Since $u^\star$ is maximizing $g(x^\star, u) - \sum_{i=1}^N v^{(i)\star} h^{(i)}(u^{(i)})$, we have

$$\begin{aligned}
&(g(x^\star, u^\star) - \textstyle\sum_{i=1}^N v^{(i)\star} h^{(i)}(u^{(i)\star})) - \\
&\quad (g(x^\star, u) - \textstyle\sum_{i=1}^N v^{(i)\star} h^{(i)}(u^{(i)})) \geq 0 \,.
\end{aligned}$$

By complimentary slackness property of the KKT point, we get $\lambda^\star (g(x^\star, u^\star) - \sum_{i=1}^N v^{(i)\star} h^{(i)}(u^{(i)\star})) = 0$ and $g(x^\star, u^\star) \leq 0$. Combining all these implies

$$\mathcal{L}(x^\star, \lambda^\star, u^\star, v^\star) - \mathcal{L}(x^\star, \lambda, u, v^\star) \geq 0 \,.$$

We next show that $\mathcal{L}(x, \lambda^\star, u^\star, v) - \mathcal{L}(x^\star, \lambda^\star, u^\star, v^\star)$ is non-negative. Note that

$$\begin{aligned}
&\mathcal{L}(x, \lambda^\star, u^\star, v) - \mathcal{L}(x^\star, \lambda^\star, u^\star, v^\star) \\
&= f(x) - f(x^\star) + \lambda^\star (g(x, u^\star) - \textstyle\sum_{i=1}^N v^{(i)} h^{(i)}(u^{(i)\star}) - \\
&\quad \lambda^\star (g(x^\star, u^\star) - \textstyle\sum_{i=1}^N v^{(i)\star} h^{(i)}(u^{(i)\star}))) \\
&= f(x) - f(x^\star) + \lambda^\star (g(x, u^\star) - g(x^\star, u^\star)) + \\
&\quad \sum_{i=1}^N \lambda^\star (v^{(i)^*} - v^{(i)}) h^{(i)}(u^{(i)\star}) \,.
\end{aligned}$$

Since $(x^\star, v^\star)$ minimizes the Lagrangian, we have

$$f(x) - f(x^\star) + \lambda^\star (g(x, u^\star) - g(x^\star, u^\star)) \geq 0 \,.$$

Similarly, using complimentary slackness condition and the fact that $h^{(i)}(u^{(i)\star}) \leq 0$, $v^{(i)} \geq 0$, and $\lambda^\star \geq 0$, it follows that $\sum_{i=1}^N \lambda^\star (v^{(i)\star} - v^{(i)}) h^{(i)}(u^{(i)\star}) \geq 0$. Now, we show that saddle point satisfies KKT conditions. Note that

$$\begin{aligned}
&\min_{x,v} \; \mathcal{L}(x, \lambda^\star, u^\star, v) = \mathcal{L}(x^\star, \lambda^\star, u^\star, v^\star) \leq \mathcal{L}(x, \lambda^\star, u^\star, v), \\
&\max_{u,\lambda} \; \mathcal{L}(x^\star, \lambda, u, v^\star) = \mathcal{L}(x^\star, \lambda^\star, u^\star, v^\star) \geq \mathcal{L}(x^\star, \lambda, u, v^\star) \,.
\end{aligned}$$

Hence,

$$\begin{aligned}
\partial_x f(x^\star) + \lambda^\star \partial_x g(x^\star, u^\star) &= 0 \,, \\
\partial_{u^{(i)}} g(x^\star, u) - \textstyle\sum_{i=1}^N v^{(i)\star} \partial_{u^{(i)}} h^{(i)}(u^{(i)\star}) &= 0 \,.
\end{aligned}$$

To show complimentary slackness, consider the optimization problem with fixed $x = x^\star$ as

$$\max_{u^{(i)},\forall i} \; g(x^\star, u) \text{ s.t. } h^{(i)}(u^{(i)}) \leq 0 \;,\; i = 1, \ldots, N \;.$$

With g concave in u and each $h^{(i)}$ convex in $u^{(i)}$, the above problem is convex with zero duality gap and hence, based on convex optimization theory[7], we have

$$\begin{aligned} g(x^\star, u^\star) &= G(x^\star, v^\star) \\ &= \max_{u^{(i)},\forall i} \left(g(x^\star, u) - \textstyle\sum_{i=1}^N v^{(i)\star} h^{(i)}(u^{(i)}) \right) \\ &\geq g(x^\star, u^\star) - \textstyle\sum_{i=1}^N v^{(i)\star} h^{(i)}(u^{(i)\star}) \geq \; g(x^\star, u^\star) \;. \end{aligned}$$

The first inequality is true because $h^{(i)}(u^{(i)\star}) \leq 0$ and $v^{(i)\star} \geq 0$. Hence, from the last inequality we get $v^{(i)\star} h^{(i)}(u^{(i)\star}) = 0$. We next show that $\lambda^\star g(x^\star, u^\star) = 0$. For fixed $u^\star$, consider the optimization problem

$$\min_x \; f(x) \;\; \text{s.t. } \; g(x, u^\star) \leq 0 \;.$$

For fixed $u^\star$, above is a convex optimization problem and hence, we have zero duality gap. Then similarly,

$$\begin{aligned} f(x^\star) = F(\lambda^\star, u^\star) &= \inf_x f(x) + \lambda^\star g(x, u^\star) \\ &\leq f(x^\star) + \lambda^\star g(x^\star, u^\star) \leq \; f(x^\star) \;. \end{aligned}$$

The first inequality is true because $g(x^\star, u^\star) \leq 0$, and hence, the last inequality implies $\lambda^\star g(x^\star, u^\star) = 0$. This completes the proof of Theorem 6. □

We can specify the equilibrium point $(x^\star, \lambda^\star, u^\star, v^\star)$ of the dynamical system (11)-(14) as

$$\begin{aligned} \partial_x f(x^\star) + \lambda^\star \partial_x g(x^\star, u^\star, \xi) &= 0, \\ \partial_{u^{(i)}} g(x^\star, u^\star, \xi) - v^{(i)\star} \partial_{u^{(i)}} h^{(i)}(u^{(i)\star}) &= 0 \;, \\ \lambda^\star g(x^\star, u^\star, \xi) &= 0, \\ \lambda^\star &\geq 0, \\ g(x^\star, \lambda^\star, \xi) &\leq 0, \\ \lambda^\star v^{(i)\star} h^{(i)}(u^{(i)\star}) &= 0, \\ v^{(i)\star} &\geq 0, \\ h^{(i)}(u^{(i)\star}) &\leq 0. \end{aligned}$$

for $i = 1, \ldots, N$. The above conditions can also be viewed as the generalization of the KKT conditions from the deterministic setting to stochastic setting. Furthermore, by defining the Lagrangian function, $\mathcal{L}(x, \lambda, u, v, \xi) := f(x) + \lambda(g(x, u, \xi) - \sum_{i=1}^N v^{(i)} h^{(i)}(u^{(i)}))$, following generalization of saddle point condition from deterministic setting (25) to stochastic setting can be considered

$$\mathcal{L}(x^\star, \lambda, u, v^\star, \xi) \leq \mathcal{L}(x^\star, \lambda^\star, u^\star, v^\star, \xi) \leq \mathcal{L}(x, \lambda^\star, u^\star, v, \xi) \;. \tag{30}$$

Convergence Proof of Stochastic Version of the Algorithm

For the convergence proof of SSDS algorithm in (11)-(14), we will assume $\lambda^\star > 0$ and that numbers R and $R_\lambda \leq R$ are known satisfying

$$\|z_1\|_2 \leq R, \; \|z^\star\|_2 \leq R, \; \|\lambda^\star\|_2 \leq R_\lambda \;.$$

We will also assume that the norm of the subgradients of f, g and $h^{(i)}$ s, and the values of f, g and $h^{(i)}$ s are bounded on compact sets based on Assumption 1. Let us start by defining the compact notations

$$\begin{aligned} \mathcal{N}(\|z_{k+1} - z^\star\|_2^2) &:= \|x_{k+1} - x^\star\|_2^2 + \|\lambda_{k+1} - \lambda^\star\|_2^2 \\ &\quad + \lambda^\star \|u_{k+1} - u^\star\|_2^2 + \|v_{k+1} - v^\star\|_2^2 \;, \\ (z_{k+1} - z^\star)_{\lambda^\star} &:= (x_{k+1} - x^\star) + (\lambda_{k+1} - \lambda^\star) \\ &\quad + \lambda^\star (u_{k+1} - u^\star) + (v_{k+1} - v^\star) \;, \\ \|T\|_{2,\lambda^\star}^2 &:= \|T^{(x)}\|_2^2 + \|T^{(\lambda)}\|_2^2 \\ &\quad + \lambda^\star \|T^{(u)}\|_2^2 + \|T^{(v)}\|_2^2 \;. \end{aligned}$$

By using the non-expansive property of positive projection operations for λ and v iterations, we write out the following basic equations

$$\begin{aligned} & E_{\xi_k}[\mathcal{N}(\|z_{k+1} - z^\star\|_2^2)] \\ &= E_{\xi_k}[\|x_k - \alpha_k(\partial_x f(x_k, \xi_k) + \lambda_k \partial_x g(x_k, u_k, \xi_k) - x^\star\|_2^2] \\ &\quad + E_{\xi_k}[\|[\lambda_k + \alpha_k g(x_k, u_k, \xi_k) - v_k h(u_k)]_+ - \lambda^\star\|_2^2] \\ &\quad + E_{\xi_k}[\lambda^\star \|u_k + \alpha_k(\partial_u g(x_k, u_k, \xi_k) - v_k \partial_u h(u_k) - u^\star\|_2^2] \\ &\quad + E_{\xi_k}[\|[v_k + \alpha_k(\lambda_k h(u_k))]_+ - v^\star\|_2^2] \\ &\leq E_{\xi_k}[\|x_k - x^\star - \alpha_k(\partial_x f(x_k, \xi_k) + \lambda_k \partial_x g(x_k, u_k, \xi_k))\|_2^2] \\ &\quad + E_{\xi_k}[\|\lambda_k - \lambda^\star + \alpha_k(g(x_k, u_k, \xi_k) - v_k h(u_k))\|_2^2] \\ &\quad + E_{\xi_k} \lambda^\star \|u_k - u^\star + \alpha_k(\partial_u g(x_k, u_k, \xi_k) - v_k \partial_u h(u_k))\|_2^2] \\ &\quad + E_{\xi_k}[\|v_k - v^\star + \alpha_k(\lambda_k h(u_k))\|_2^2] \\ &= \|x_k - x^\star\|_2^2 + \|\lambda_k - \lambda^\star\|_2^2 + \lambda^\star \|u_k - u^\star\|_2^2 + \|v_k - v^\star\|_2^2 \\ &\quad - 2E_{\xi_k}[\alpha_k(\partial_x f(x_k, \xi_k) + \lambda_k \partial_x g(x_k, u_k, \xi_k))^\top (x_k - x^\star)] \\ &\quad + 2E_{\xi_k}[\alpha_k(g(x_k, u_k, \xi_k) - v_k h(u_k))^\top (\lambda_k - \lambda^\star)] \\ &\quad + 2E_{\xi_k}[\alpha_k \lambda^\star(\partial_u g(x_k, u_k, \xi_k) - v_k \partial_u h(u_k))^\top (u_k - u^\star)] \\ &\quad + E_{\xi_k}[2\alpha_k(\lambda_k h(u_k))^\top (v_k - v^\star)] \\ &\quad + E_{\xi_k}[(\alpha_k)^2 \|\partial_x f(x_k, \xi_k) + \lambda_k \partial_x g(x_k, u_k, \xi_k)\|_2^2] \\ &\quad + E_{\xi_k}[(\alpha_k)^2 \|g(x_k, u_k, \xi_k) - v_k h(u_k)\|_2^2] \\ &\quad + E_{\xi_k}[(\alpha_k)^2 \lambda^\star \|\partial_u g(x_k, u_k, \xi_k) - v_k \partial_u h(u_k)\|_2^2] \\ &\quad + E_{\xi_k}[(\alpha_k)^2 \|\lambda_k h(u_k)\|_2^2] \;. \end{aligned}$$

Using the compact notation, this reads to be

$$E_{\xi_k}[\mathcal{N}(\|z_{k+1}-z^\star\|_2^2)] \le \mathcal{N}(\|z_k-z^\star\|_2^2) - 2E_{\xi_k}[\alpha_k T_k^\top \mathcal{N}(z_{k+1}-z^\star)] + E_{\xi_k}[\alpha_k^2 \|T_k\|_{2,\lambda^\star}^2] .$$

Considering the upper bound R_λ for two-norm of $\lambda^\star$, we can write

$$E_{\xi_k}[\mathcal{N}(\|z_{k+1}-z^\star\|_2^2)] \le \mathcal{N}(\|z_k-z^\star\|_2^2) - 2E_{\xi_k}[\alpha_k T_k^\top \mathcal{N}(z_{k+1}-z^\star)] + C\,\gamma_k^2 .$$

Taking expectation on both the sides with respect to $E_{\xi_0^{k-1}}$ on both the sides and using the fact that ξ_0^{k-1} is independent of ξ_k, we obtain

$$E_{\xi_0^k}[\mathcal{N}(\|z_{k+1}-z^\star\|_2^2)] \le E_{\xi_0^{k-1}}\mathcal{N}(\|z_k-z^\star\|_2^2) - 2E_{\xi_0^k}[\alpha_k T_k^\top \mathcal{N}(z_{k+1}-z^\star)] + C\,\gamma_k^2 ,$$

$$E_{\xi_0^{k-1}}[\mathcal{N}(\|z_k-z^\star\|_2^2)] \le E_{\xi_0^{k-2}}\mathcal{N}(\|z_{k-1}-z^\star\|_2^2) \tag{31}$$

$$- 2E_{\xi_0^{k-1}}[\alpha_{k-1} T_{k-1}^\top \mathcal{N}(z_k-z^\star)] + C\,\gamma_{k-1}^2, \tag{32}$$

where C is defined as $\max\{1, R_\lambda\}$. Substituting inequality (9) into (8) we obtain

$$E_{\xi_0^k}[\mathcal{N}(\|z_{k+1}-z^\star\|_2^2)] \le E_{\xi_0^{k-2}}\mathcal{N}(\|z_{k-1}-z^\star\|_2^2) - 2E_{\xi_0^{k-1}}[\alpha_{k-1} T_{k-1}^\top \mathcal{N}(z_k-z^\star)] - 2E_{\xi_0^k}[\alpha_k T_k^\top \mathcal{N}(z_{k+1}-z^\star)] + C(\gamma_k^2+\gamma_{k-1}^2) .$$

Using recursion, we obtain

$$E_{\xi_0^k}[\mathcal{N}(\|z_{k+1}-z^\star\|_2^2)] \le E_{\xi_0}\mathcal{N}(\|z_1-z^\star\|_2^2) - 2\sum_{i=1}^k E_{\xi_0^i}[\alpha^{(i)} T^{(i)\top}\mathcal{N}(z_{i+1}-z^\star)] + C\sum_{i=1}^k \gamma^{(i)^2} ,$$

$$E_{\xi_0^k}[\mathcal{N}(\|z_{k+1}-z^\star\|_2^2)] + 2\sum_{i=1}^k E_{\xi_0^i}[\alpha^{(i)} T^{(i)\top}\mathcal{N}(z_{i+1}-z^\star)] \le E_{\xi_0}\mathcal{N}(\|z_1-z^\star\|_2^2) + C\sum_{i=1}^k \gamma^{(i)^2} \le C(4R^2+S)$$

$$E_{\xi_0^k}[\mathcal{N}(\|z_{k+1}-z^\star\|_2^2)] + 2\Big(E_{\xi_0^1}[\alpha_1 T_1^\top \mathcal{N}(z_2-z^\star)] + E_{\xi_0^2}[\alpha_2 T_2^\top \mathcal{N}(z_3-z^\star)] + \ldots + E_{\xi_0^k}[\alpha_k T_k^\top \mathcal{N}(z_{k+1}-z^\star)]\Big) \le E_{\xi_0}\mathcal{N}(\|z_1-z^\star\|_2^2) + C\sum_{i=1}^k \gamma^{(i)^2} \le C(4R^2+S) , \tag{33}$$

where the last inequality comes from the bounds on $\|z_1\|_2$, $\|z^\star\|_2$, $\|\lambda^\star\|_2$ and $\sum_{k=1}^\infty (\gamma_k)^2$.

We argue that the sum on the left-hand side of (33) is non-negative.

$$E_{\xi_0^k}[\alpha_k T_k^\top \mathcal{N}(z_{k+1}-z^\star)] = E_{\xi_0^{k-1}}[[E_{\xi_k}[\alpha_k T_k^\top \mathcal{N}(z_{k+1}-z^\star)]]]$$

Where we have use the fact that ξ_0^{k-1} is independent of ξ_k.

$$\begin{aligned}
&E_{\xi_k}[\alpha_k T_k^\top \mathcal{N}(z_{k+1}-z^\star)] \\
&= E_{\xi_k}[\alpha_k \partial_x f(x_k,\xi_k) + \alpha_k\lambda_k \partial_x g(x_k,u_k,\xi_k)^\top (x_k-x^\star)] \\
&\quad - E_{\xi_k}[\alpha_k (g(x_k,u_k,\xi_k) - v_k h(u_k))^\top (\lambda_k-\lambda^\star)] \\
&\quad - E_{\xi_k}[\alpha_k\lambda^\star (\partial_u g(x_k,u_k,\xi_k) - v_k \partial_u h(u_k))^\top (u_k-u^\star)] \\
&\quad + E_{\xi_k}[-\alpha_k\lambda_k h(u_k)^\top (v_k-v^\star)] \\
&\ge E_{\xi_k}[\alpha_k (f(x_k,\xi_k) - f(x^\star,\xi_k))] + E_{\xi_k}[\alpha_k\lambda_k g(x_k,u_k,\xi_k) \\
&\quad - \alpha_k\lambda_k g(x^\star,u_k,\xi_k)] - E_{\xi_k}[\alpha_k\lambda_k g(x_k,u_k,\xi_k) \\
&\quad + \alpha_k\lambda^\star g(x_k,u_k,\xi_k)] + E_{\xi_k}[\alpha_k\lambda_k v_k h(u_k) - \alpha_k\lambda^\star v_k h(u_k)] \\
&\quad + E_{\xi_k}[\alpha_k\lambda^\star g(x_k,u^\star,\xi_k) - \alpha_k\lambda^\star g(x_k,u_k,\xi_k)] \\
&\quad + E_{\xi_k}[\alpha_k\lambda^\star v_k h(u_k)] - E_{\xi_k}[\alpha_k\lambda^\star v_k h(u^\star) - \alpha_k\lambda_k v_k h(u_k) \\
&\quad + \alpha_k\lambda_k v^\star h(u_k)] \\
&= E_{\xi_k}[\alpha_k((f(x_k,\xi_k) + \lambda^\star g(x_k,u^\star,\xi_k) - \lambda^\star v_k h(u^\star))] \\
&\quad - E_{\xi_k}[\alpha_k(f(x^\star,\xi_k) + \lambda_k g(x^\star,u_k,\xi_k) - \lambda_k v^\star h(u_k))] \\
&= E_{\xi_k}[\alpha_k \mathcal{L}(x_k,\lambda^\star,u^\star,v_k,\xi_k)] - E_{\xi_k}[\alpha_k \mathcal{L}(x^\star,\lambda_k,u_k,v^\star,\xi_k)] \\
&\ge E_{\xi_k}[\alpha_k \mathcal{L}(x_k,\lambda^\star,u^\star,v_k,\xi_k) - \alpha_k \mathcal{L}(x^\star,\lambda^\star,u^\star,v^\star,\xi_k)] \ge 0 .
\end{aligned}$$

Since $\alpha_k \ge 0$, we have from above that

$$E_{\xi_0^k}[\alpha_k T_k^\top \mathcal{N}(z_{k+1}-z^\star)] \ge 0$$

Remark 4. *Since f is assumed to be strictly convex in x and g is strictly concave in u, the above inequality is strict whenever $x \ne x^\star$ and $u \ne u^\star$. Moreover, if the inequality becomes an equality, we get $x = x^\star$ and $u = u^\star$.*

We have

$$E_{\xi_0^k}[\mathcal{N}(\|z_{k+1}-z^\star\|_2^2)] \le C(4R^2+S),$$
$$2\sum_{i=1}^k \gamma^{(i)} E_{\xi_0^i}\left[\frac{T^{(i)\top}}{\|T^{(i)}\|}(z^{(i)}-z^\star)_{\lambda^\star}\right] \le C(4R^2+S)$$

By assumption, the norm of Subgradients on the set $\|z_k\|_{2,\lambda^\star} \le D$ is bounded, so it follows that $\|T_k\|_2$ is bounded. Because the sum of γ_k diverges, for the sum

$$\sum_{i=1}^k \gamma^{(i)} E_{\xi_0^i}\left[\frac{T^{(i)\top}}{\|T^{(i)}\|}(z_{i+1}-z^\star)_{\lambda^\star}\right]$$

to be bounded, we need

$$\lim_{k\to\infty} E_{\zeta_0^k}\left[\frac{T_k^\top}{\|T_k\|_2}\mathcal{N}(z_{k+1}-z^\star)\right]=0.$$

Since $\|T_k\|_2$ is bounded, the numerator $E_{\zeta_0^k}[T_k^\top\mathcal{N}(z_{k+1}-z^\star)]$ has to go to zero in the limit. From Remark 4, we conclude that

$$\lim_{k\to\infty} E_{\zeta_0^{k-1}}[x_k]=x^\star,\quad \lim_{k\to\infty} E_{\zeta_0^{k-1}}[u_k]=u^\star.$$

Experimental Results

In this section, more elaborations about SSDS algorithm will be provided, and different experiments using SSDS-p algorithm will be discussed.

SSDS-p Model Convergence Results

As discussed in the main paper, although SSDS can control the perturbations to remain under the budget for most of the data points, we add a projection term for the u update in order to make sure that we are always perturbing within the budget. Fig. 5 shows the accuracy and loss value plots for SSDS-p training. Also, as shown in Fig. 7, for SSDS-p algorithm $\|u\|_\infty$ never goes beyond the budget and the ℓ_∞ norm of the final perturbations stays at the specified budget (0.03 here) for the majority of the images in the dataset (see Fig. 5c). Similar to the observation we made for SSDS case, v converges shortly after the evolution starts (at epoch 180 here), but the dynamics for finding the desired attack, $\|u_k-u_{K-1}\|_2$ converges to zero later (epoch 250 here). Fig. 7a shows how the natural image, final attack, and the final corrupted image look like.

SSDS and SSDS-p attacks

As discussed in Section 1, SSDS attacks are found through an iterative process by freezing the model ($w^\star$) and $\lambda^\star$ and using the update rules for u and v. Therefore, we follow the steps discussed in algorithm 2 for SSDS attacks. The notion holds for SSDS-p attacks except for the projection term which keeps the attacks within the budget at all times.

Fig. 8 shows the performance of the different models *i.e.*, SSDS-p, PGD, FGSM, and naturally trained models on SSDS-p attacks. The final values are reported in Table 1, and as the plots are showing, the accuracy is high in the beginning when the algorithm hasn't still converged and the final perturbation is still not found, but as the final perturbation is found, the accuracy stays fixed at the testing accuracy of the corresponding model on SSDS-p attack.

Algorithm 2 SSDS attack algorithm

1: **Input**: ε, p, C_1, $w^\star$
2: **Initialization**:u_0, v_0, α_0
3: **for** $k\in\{1,...,K\}$ **do**
4: $\quad v_{k+1}^{(m_j)}=v_k^{(m_j)}+\alpha_k(\|u_k^{(m_j)}\|_\infty-\varepsilon)$
5: $\quad u_{k+1}^{(m_j)}=u_k^{(m_j)}+\alpha_k(\partial_{u_k}L(I^{(m_j)}+u_k^{(m_j)},y^{(m_j)},w^\star)-C_1\,v_k^{(m_j)}\,\mathrm{sign}(u_k^{(m_j)}))$
6: $\quad \alpha_{k+1}=\alpha_k e^{-kp}$
7: **end for**

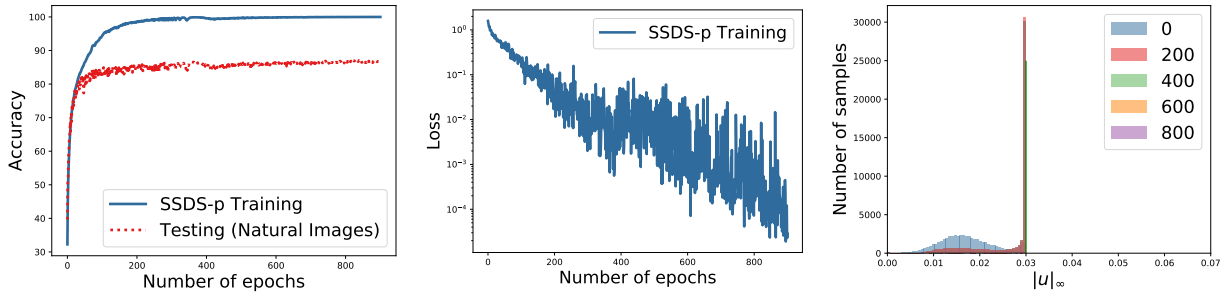
Fig. 9 shows the visual differences between these three attacks on a randomly selected image. The clean image(left), the corresponding corruption(center) and the corrupted image (right) are provided for each method. Figs. 9b and 9a show that the notion of imperceptibility holds for SSDS and PGD attacks, where as FGSM perturbations are much less imperceptible.

Evolution of the attacks

Based on what we discussed in previous chapters, the SSDS algorithm goes through an iterative process for finding the adversary. Therefore, the attack pattern evolves significantly as the algorithm progresses. As shown in Fig. 10, the attack pattern changes until the final attack is found, and then it remains almost the same after the algorithm converges (here at epoch 260).

VGG model evaluation Under transfer attacks:

We note that for all the white-box results presented in Table 3, we use a VGG19 [15] model architecture with the same set of hyper-parameters. However, Madry *et al*. [18] used a wide Resnet model[31] trained with a very large number of epochs (80000 epochs) to achieve the state-of-the-art robust models. Therefore, we also tested our SSDS-p robust model on the corrupt test cases generated by Madry *et al*. In this regard, the accuracy values reported in the first 3 columns of Table 4 are directly borrowed from[18]. Our SSDS-p trained model (still a VGG-19 model) demonstrates a strong performance on these benchmark data sets. However, these attacks are transfer attacks for the SSDS-p model while they are white box attacks for the other models listed here. Therefore, SSDS-p model is expected to perform slightly better in these cases[21]. More interestingly, we find that the SSDS-p model performs significantly better



(a) (b) (c)

Figure 5 *Results for training CIFAR-10 dataset using SSDS-p algorithm, VGG19 model architecture: (a) Accuracy (b) Loss value (log scale) (c)Histogram of* $\|u\|_\infty$

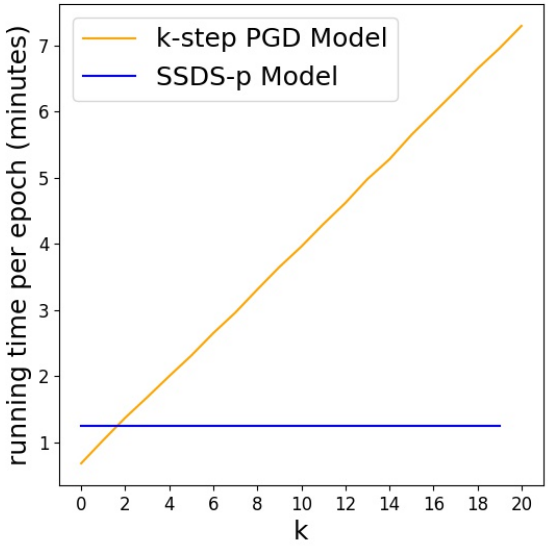


Figure 6 *Training time per epoch for PGD vs. SSDS-p (VGG19 model architecture)*

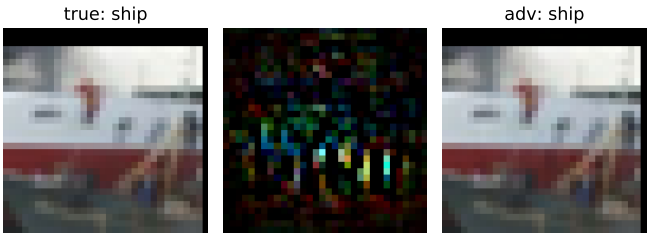
than the wide PGD model in a pure black-box setting. While the wide PGD model has a black-box accuracy of **64.2**% under a 7 step PGD attack as reported in [18], SSDS-p has an accuracy of **69.58**%. Note that the model used in this experiment is trained for longer time (3000 epochs) whereas the model used in white-box evaluation experiment (Table 3) is trained for 1000 epochs. Therefore, the test accuracy of SSDS-P model on clean images is slightly higher compared to the same accuracy reported in Table 3.

Table 4 *Performance of different models on Madry et al. [18] generated test cases*

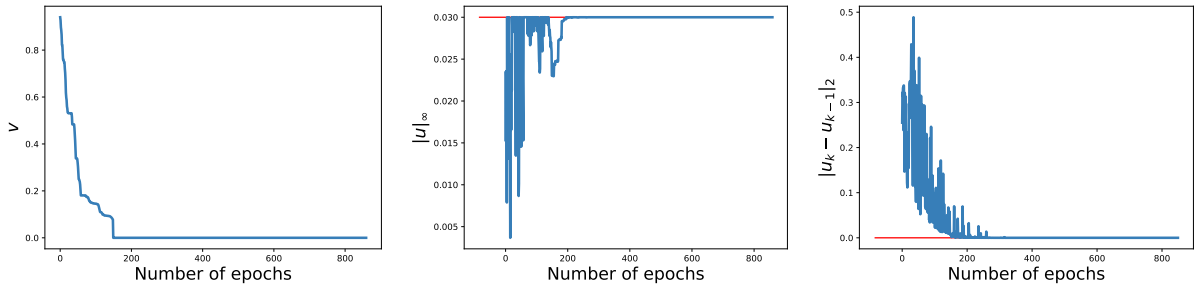
Attack \ Model	Natural training	FGSM training	PGD training	SSDS-p
Clean	92.70%	87.40%	79.04%	86.83%
FGSM	27.5%	90.09%	51.70%	70.55%
PGD	0.80%	0.00%	43.70%	67.52%

Training time comparison for VGG model architecture

Similar to what we did for the Resnet50 model, We compare the running time for SSDS-p and k-step PGD for our VGG19 model. Fig. 6 shows that the training time per epoch for 20 step PGD-training algorithm is approximately 7 times greater than that of the SSDS-p training algorithm. On the other hand, SSDS-p training takes around the same time as a 2 step PGD training.

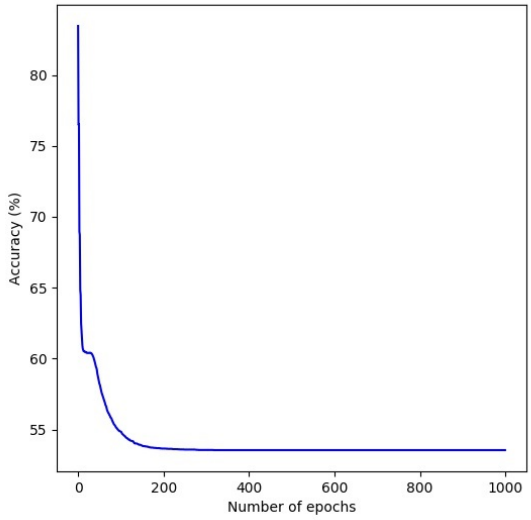


(a)

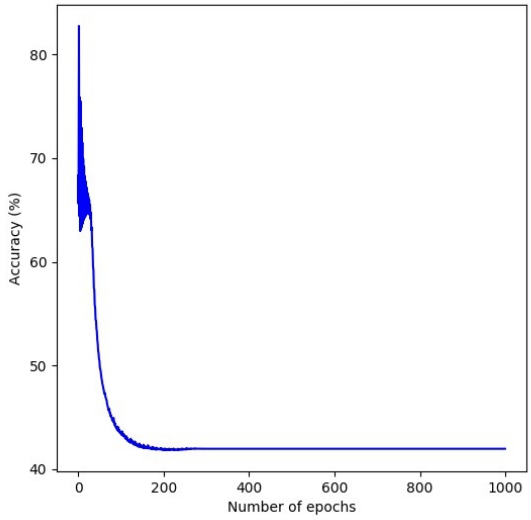


(b) **(c)** **(d)**

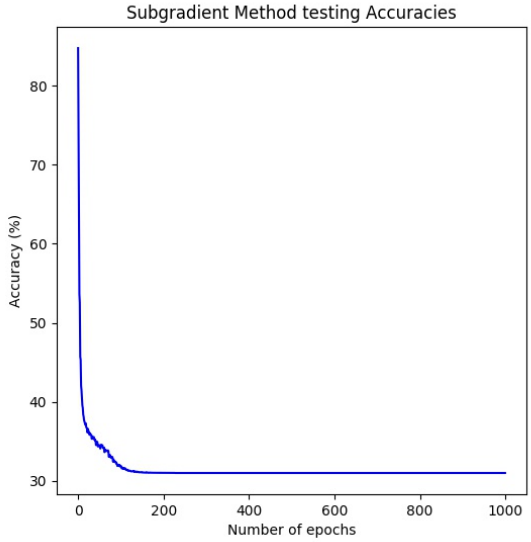
Figure 7 *Results for training CIFAR-10 dataset, SSDS-p algorithm, VGG19 model architecture:(a) Example of SSDS-p training image(left), with its corruption(center) and the corrupted image(right), (b) v values for the above image (c) $\|u\|_\infty$ for the above image, (d)$|u_k - u_{K-1}\|_2$ for the above image*



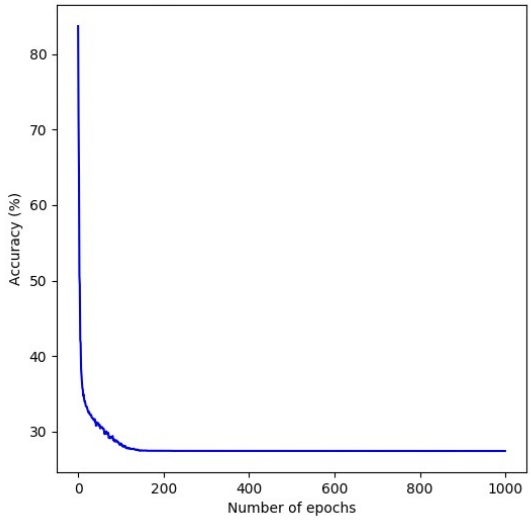
(a) *SSDS-p model*



(b) *PGD model*

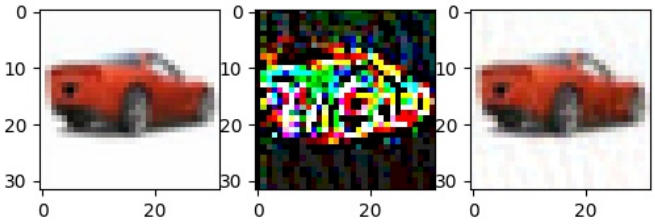


(c) *FGSM model*

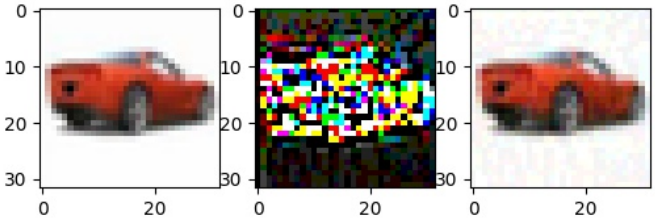


(d) *Natural model*

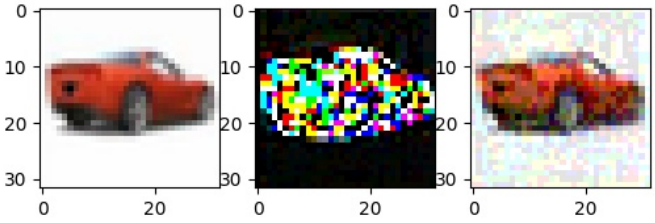
Figure 8 *Accuracy plots of different models on SSDS-p attacked images with* $\varepsilon = 0.03$ *budget*



(a) *SSDS Attack*

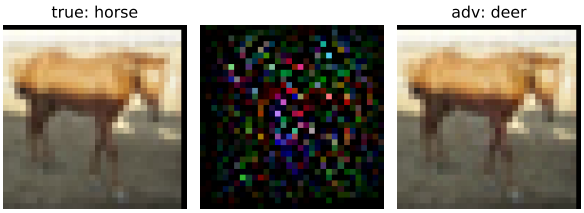


(b) *PGD attack*

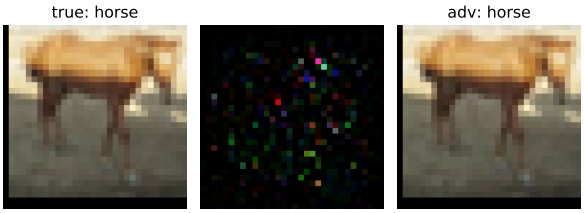


(c) *FGSM attack*

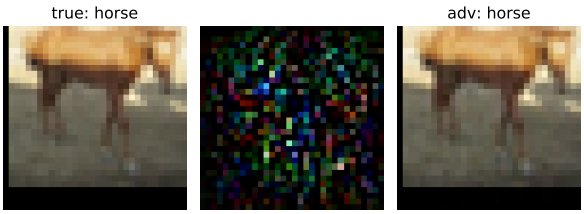
Figure 9 *Different attack model visualizations*



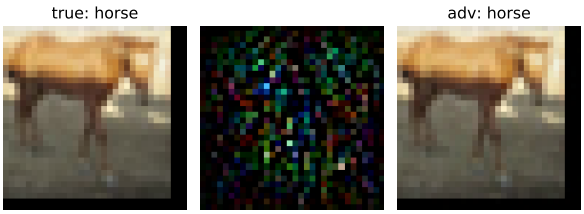
(a) *50th epoch*



(b) *180th epoch*



(c) *260th epoch*



(d) *700th epoch*

Figure 10 *SSDS attack evolution*