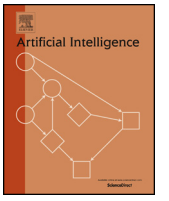
Artificial Intelligence 315 (2023) 103837

Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint



Recursive reasoning-based training-time adversarial machine learning ☆

Yizhou Chen, Zhongxiang Dai, Haibin Yu, Bryan Kian Hsiang Low *, Teck-Hua Ho

National University of Singapore, 21 Lower Kent Ridge Road, 119077, Singapore

ARTICLE INFO

Article history:
Received 12 January 2022
Received in revised form 30 November 2022
Accepted 6 December 2022
Available online xxxx

Keywords:
Recursive reasoning
Adversarial machine learning
Game theory

ABSTRACT

The training process of a *machine learning* (ML) model may be subject to adversarial attacks from an attacker who attempts to undermine the test performance of the ML model by perturbing the training minibatches, and thus needs to be protected by a defender. Such a problem setting is referred to as training-time adversarial ML. We formulate it as a two-player game and propose a principled *Recursive Reasoning-based Training-Time adversarial ML* (R2T2) framework to model this game. R2T2 models the reasoning process between the attacker and the defender and captures their bounded reasoning capabilities (due to bounded computational resources) through the recursive reasoning formalism. In particular, we associate a deeper level of recursive reasoning with the use of a higher-order gradient to derive the attack (defense) strategy, which naturally improves its performance while requiring greater computational resources. Interestingly, our R2T2 framework encompasses a variety of existing adversarial ML methods which correspond to attackers (defenders) with different recursive reasoning capabilities. We show how an R2T2 attacker (defender) can utilize our proposed nested projected gradient descent-based method to approximate the optimal attack (defense) strategy at an arbitrary level of reasoning. R2T2 can empirically achieve state-of-the-art attack and defense performances on benchmark image datasets.

1. Introduction

The widespread use of *machine learning* (ML) models has inevitably raised concerns about the security and safety of their application in the real world, which has galvanized the recent popularity of *adversarial ML*. In particular, the training process of an ML model may be subject to adversarial attacks whose objective is to fool the ML model into performing well during training yet inadequately during deployment. Therefore, a defense mechanism is needed to ensure the reliability of the ML model.

An important motivating scenario is an online learning system in which the model learning performed by a central server requires training data that are sent in minibatches from some data source(s). This is an increasingly prevalent setting in modern ML where large amounts of data are available [1,2]. Due to memory constraints, the data is usually gathered and

☆ This paper is part of the Special Issue: Risk-aware Autonomous Systems: Theory and Practice.

* Corresponding author.
E-mail addresses: leon798775190@gmail.com (Y. Chen), daiz9109@gmail.com (Z. Dai), buaanusyu@gmail.com (H. Yu), lowkh@comp.nus.edu.sg (B.K.H. Low), teck@nus.edu.sg (T.-H. Ho).

https://doi.org/10.1016/j.artint.2022.103837
0004-3702/

stored locally, and then transmitted to the central server for model update. This system is susceptible to attacks from (a) an adversarial data provider who sends out poisonous minibatch data or (b) a malicious hacker who can access and modify the transmitted data. These attackers usually attempt to compromise the test-time performance of the trained ML model on clean data samples. As a countermeasure, the central server can invoke a defender to correct the received data before using them for model update.

Formally, we refer to this problem setting as *training-time adversarial ML* which can be formulated as a *two-player game* between the *training-time attacker* and *training-time defender*. In such a game, it is usually too complicated to solve for a Nash equilibrium and hence not realistic for both players to employ Nash equilibrium strategies.[1] Nevertheless, an interesting research question remains to be answered: *How should the attacker (defender) choose its strategy to achieve more effective attacks (defenses) and thus gain an advantage in the game?*

Unfortunately, most works on adversarial ML have (a) focused on test-time attacks/defenses and (b) studied either attacks or defenses solely. Test-time attacks [4] and defenses [5] are well-studied. However, as we will show in our experiments, they do not transfer well to training-time attacks and defenses. The work of [6] has proposed learning an auto-encoder-like network to generate training-time attacks without considering defenses. To our best knowledge, a detailed study of training-time defenses is still lacking.

In training-time adversarial ML, the training of the ML model (referred to as the *target ML model* hereafter) is affected by the strategies of both the attacker and defender. So, for the attacker to gain an advantage in the game, it has to reason about the defender's strategy. Similarly, to derive a more effective defense strategy, the defender can adopt a reasoning process that accounts for the attacker's potential strategy and belief about the defender. This gives rise to the *recursive reasoning* process which is captured by the *level K* (LK) model from behavioral game theory for explaining the behavior of humans [7–10]. In the LK model, every player (person) reasons at a particular level: A level-0 player selects a null strategy while a level-$k \geq 1$ player best-responds to the level-$(k-1)$ behavior of the opponent.

A person with a more sophisticated strategic thinking is capable of a deeper level of reasoning and usually performs better in a game. Similarly, in the game of training-time adversarial ML, a computational agent (attacker/defender) with access to more computational resources is expected to be able to find a better attack/defense strategy. In behavioral game theory, a person's reasoning level is determined by her cognitive capability, otherwise known as *cognitive limit* [11–13]. This naturally brings up a question: How should we characterize the cognitive limit of a computational agent with bounded computational resources and thus determine its reasoning level? In this work, we associate the cognitive limit and hence reasoning level of an attacker/defender with the *highest order of gradient* that can be computed by the player. This association turns out to be natural and elegant since having access to more gradient information is likely to improve its performance and the cost of computing a gradient grows exponentially in its order.

This paper presents the first principled *Recursive Reasoning-based Training-Time adversarial ML* (R2T2) framework (Sec. 4) to model training-time adversarial ML as a 2-player game. Interestingly, our R2T2 framework encompasses a variety of existing adversarial ML methods which correspond to attackers (defenders) with different recursive reasoning capabilities. We show how an R2T2 attacker (defender) can utilize our proposed nested projected gradient descent-based method to approximate the optimal attack (defense) strategy at an arbitrary level of reasoning (Sec. 4.4). We empirically demonstrate that R2T2 can achieve state-of-the-art attack and defense performances on benchmark image datasets (Sec. 6) and, for the first time, provide principled guidelines to protect the training of an ML model against training-time attacks.

2. Related work

Most works on adversarial ML have focused on *test-time attacks* [4,14–16] where the attacker tries to fool an already trained ML model into an incorrect prediction by perturbing the test input. Several works aim at defending against such test-time attacks through data sanitization by correcting the perturbed test input [17–19]. Note that all these works have studied either attacks or defenses solely. Their straightforward application to training-time attacks/defenses do not yield satisfactory performances, as demonstrated in our experiments (Sec. 6).

Another line of work called *adversarial training* [5] has attempted to perform test-time defense by including a training-time attacker during the training of the ML model. As a result, these attacks during training can act as a "vaccine" to ensure that the resulting model is robust against similar test-time attacks [4,5,15,20–22]. In adversarial training, the attacker is playing against a level-0/null-strategy defender under our R2T2 framework (Sec. 3.1).

Some other works have studied the training of an ML model under *dataset poisoning*, that is, by modifying the training set *before* the training starts and only allowing a few data points to be changed [23,24]. In comparison, we assume that the attacker can access and modify the minibatch data *during* model training. The dataset poisoning method of [25] makes use of influence function and can be considered a variant of our level-2 attack strategy (Sec. 4.3). A number of works have been proposed to defend against a dataset poisoning attack based on outlier removal [26–28], but they do not fit into our setting where an attack can corrupt all inputs of a minibatch. Some of the works on poisoning attacks and defences [28–30] have formulated a Stackelberg game between the attacker and the defender, and then proposed solutions based on bilevel

[1] The work of [3] studies Nash Equilibrium strategies under the assumption of a binary classifier and locally linear decision boundaries.

optimization. In comparison, our work can also be viewed as a Stackelberg game where the attacker is the leader and the defender is the follower. Then, our optimal strategies correspond to a *sub-game perfect Nash equilibrium* (SPNE).

The works on *backdoor attacks* [31,32] have tried injecting a backdoor into a model during training to produce wrong predictions if a specific trigger is added to an input at test time. Such attacks, which require the test set to be chosen/manipulated to be backdoored tasks, do not fit into our setting that does not allow modifying the test set. Backdoor defenses like backdoor removal [33] are ineffective in our setting where an attack does not inject any backdoor.

The work of [34] has proposed to use recursive reasoning in multi-agent Bayesian optimization but did not associate the level of recursive reasoning with any specific information (e.g., order of gradient in our case). This work has illustrated its application to test-time attacks, while we focus on training-time adversarial ML here.

Most relevant to our work is that of [6] learning an auto-encoder-like network to generate training-time attacks, which essentially amounts to learning a variant of our level-1 attack strategy (Sec. 4.2). We show in Sec. 4.5 how our R2T2 framework encompasses a variety of existing adversarial ML methods which correspond to attackers/defenders with different reasoning levels.

3. Background and notations

Attack Strategy. Suppose that the target ML model is to be trained for a total of T iterations. In each training iteration $t = 0, 1, \ldots, T-1$, the attacker can intercept the minibatch data $\mathcal{D}_t$ transmitted to the ML model.[2] Let $\mathcal{D}_t^X$ denote the *inputs* of the minibatch $\mathcal{D}_t$. An *attack strategy* δ_t is a function mapping each input $x_t \in \mathcal{D}_t^X$ to a perturbation $\delta_t(x_t)$ that is added back to x_t to yield the perturbed input:

$$x_t' \triangleq x_t + \epsilon_{\text{atk}}\, \delta_t(x_t) \tag{1}$$

s.t. $\|\delta_t(x_t)\| \leq 1$ which can be satisfied using a function, denoted by $\vec{\mathbf{e}}(\cdot)$, that outputs the unit vector $\delta_t(x_t)$ of its unnormalized perturbation.[3]

It is realistic to assume the use of a *semantic checker* to perform a sanity check on the data prior to model training. For example, in image classification tasks, a human checker can sample training minibatches to be inspected directly, which helps to ensure that the quality of the training data is not compromised. To bypass such a semantic checker, the attacker needs its perturbed image input to be visually indistinguishable to the human eye, which amounts to setting a small ϵ_{atk}. Otherwise, a simple sanity check from the semantic checker can uncover the existence of the attacker and calls an end to the model training.

Now, suppose that the semantic checker has not detected any anomaly. Then, the model should (only) beware of attacks that have a bounded perturbation indistinguishable to the human eye.

We would also like to point out that having full control of training data (i.e., changing an entire batch of data instead of changing a few datapoints) is a realistic assumption: Recent works have studied a more practical scenario of attacking real-time data streaming [6,35,36] where the model is updated on user feedback or newly captured images. In this case, the attacker can interact with the training process and dynamically perturb every data batch according to the model states. The attackers from these works may have different objective functions from ours, but all of them at least have the above-mentioned abilities. In fact, some attackers have adopted even stronger assumptions: For example, the work of [35] assumes the attacker exactly knows all future mini-batch data.

Defense Strategy. In training iteration t, the defender receives the corrupted minibatch data containing perturbed inputs x_t''s from the attacker and cannot access its original inputs x_t's.[2] A *defense strategy* σ_t is a function mapping each perturbed input x_t' to a transformation $\sigma_t(x_t')$ that is added back to x_t' to yield the transformed input:

$$x_t'' \triangleq x_t' + \epsilon_{\text{def}}\, \sigma_t(x_t') \tag{2}$$

s.t. $\|\sigma_t(x_t')\| \leq 1$.[3] Finally, x_t'' is used by the target ML model on the central server to perform the model update for the current training iteration t. In (1) and (2), ϵ_{atk} and ϵ_{def} are pre-defined, fixed scale constants determined by external factors. As an example, an attacker needs its perturbed image input to be visually indistinguishable to the human eye in order to bypass the semantic checker. On the other hand, the defender knows that the training data, which have bypassed the semantic checker, can only carry bounded perturbations and thus will limit its own transformation by a small ϵ_{def} to prevent overfitting. Sec. 6.1 provides a further discussion. For simplicity, we assume $\epsilon \triangleq \epsilon_{\text{atk}} = \epsilon_{\text{def}}$ in our theoretical analysis.

We assume both players know that the target ML model is trained using the widely used *stochastic gradient descent* (SGD) method which is an iterative 1st-order optimization scheme.[4] In training iteration t, the model parameters θ_t are updated via SGD using the minibatch $\mathcal{D}_t$ with step size η:

[2] It is thus implicitly assumed that both players do not have access to the entire training dataset.

[3] Let $\|\cdot\|$ denote the L_2 norm which is chosen to simplify the theoretical analysis. Some previous works [4,5] use the L_∞ norm instead: $\|\delta_t(x_t)\|_\infty \leq 1$ can be easily adapted from $\|\delta_t(x_t)\| \leq 1$ by replacing $\vec{\mathbf{e}}(\cdot)$ with the sign function.

[4] In practice, using a momentum-based optimizer can better ensure training convergence (Appendix A.1.1).

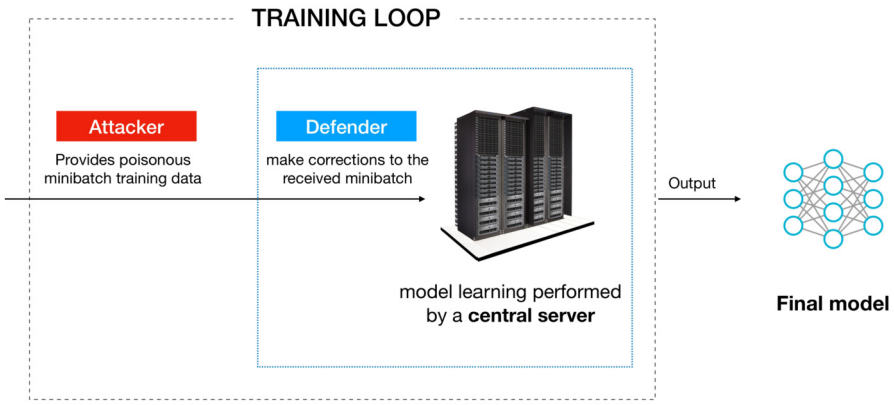


Fig. 1. The illustration of the interaction between the attacker and the defender.

$$\theta_{t+1} = \theta_t - \eta \, \nabla_\theta \sum_{x_t \in \mathcal{D}_t^X} \mathcal{L}_{\theta_t}(x_t'') \tag{3}$$

where the loss function $\mathcal{L}_\theta(x)$ denotes the loss incurred on input x by using the ML model with parameters θ to predict the ground-truth label $y(x)$ and is assumed to be smooth (i.e., infinitely differentiable) w.r.t. θ and x. We focus on **white-box attacks** and **white-box defenses**, that is, in each training iteration t, both players know the current parameters θ_t of the target ML model. Nevertheless, the attacker can realize **black-box attacks** by training a local surrogate model which follows the common practice in black-box attacks [37]. On the other hand, since the defender is with the central server, we assume that it always performs white-box defenses. More details of generalization to black-box attacks are provided in Sec. 4.6 and we also provide experimental results in Sec. 6.

Utility Functions of the Players. Following the work of [6], we assume that the goal of the training-time attacker is to misguide the model into performing poorly on the clean data samples during test time, given that the model training converges. Meanwhile, the training-time defender tries to guide the model into converging properly in the presence of the attacker to ensure a good test-time performance. We focus on *non-targeted* attacks, that is, the attacker's objective is to cause an incorrect prediction. This is in contrast to *targeted* attacks in which the attacker intends to misguide the model into producing a specific output label. Nevertheless, the extension to targeted attacks is provided in Sec. 6.4 with experimental results.

The game between the non-targeted attacker and the defender can be formulated as a zero-sum *extensive-form game* which consists of a number of repetitions of some *base games*, as illustrated in Fig. 1. Every base game corresponds to one iteration of the model training, in which both players modify the minibatch data once. Given that the test set is not known, the attacker tries to maximize the following utility (empirical risk) which the defender attempts to minimize and is a sum of base game utilities over T iterations:

$$\sum_{t=0}^{T-1} \mathcal{U}_t \quad \text{s.t.} \quad \mathcal{U}_t \triangleq \sum_{z \in \mathcal{D}_{\text{val}}^X} \left[\mathcal{L}_{\theta_{t+1}}(z) - \mathcal{L}_{\theta_t}(z) \right] \tag{4}$$

where $\mathcal{D}_{\text{val}}^X$ denotes the *inputs* of the validation set $\mathcal{D}_{\text{val}}$. To understand (4), $\mathcal{U}_t$ is the *increment* in the validation loss in iteration t while its sum over T iterations is equivalent to the final validation loss after training is completed. Unfortunately, directly solving this extensive-form game in (4) is infeasible mainly due to the following two reasons: (a) The strategy spaces are infinite. (b) In iteration t, both players may not have access to the future minibatches in iterations $t+1, \ldots, T-1$, which makes it difficult to reason about the future utilities $\mathcal{U}_{t'}$ for $t' = t+1, \ldots, T-1$. These issues hinder the use of existing approaches (such as counterfactual regret minimization and backward induction) to solve this extensive-form game. Therefore, we adopt a more practical solution by disentangling the game across different iterations. That is, we assume that in every iteration t, both players greedily optimize the base game utilities $\mathcal{U}_t$.

To simplify notations, we assume each minibatch to be of size one, i.e., $\mathcal{D}_t^X = \{x_t\}$ (our analysis can be trivially generalized to each minibatch being of size more than one, as shown in Appendix B.4). Note that in iteration t, θ_t is known and $\mathcal{U}_t$ depends on x_t'' through θ_{t+1} following the model update (3). Therefore, we denote it as a function of x_t'': $\mathcal{U}_t(x_t'')$. The greedy approach solves the following base game in iteration t, which is also zero-sum:

$$\begin{aligned} &\textbf{Attacker:} \quad \max\nolimits_{\delta_t(x_t):\|\delta_t(x_t)\| \le 1} \; \mathcal{U}_t(x_t'') \;, \\ &\textbf{Defender:} \quad \max\nolimits_{\sigma_t(x_t'):\|\sigma_t(x_t')\| \le 1} \; -\mathcal{U}_t(x_t'') \;. \end{aligned} \tag{5}$$

Note that the game between the attacker and the defender is essentially a Stackelberg game where the attacker is the leader and the defender is the follower. Computing the optimal strategy of a particular player in (5) is equivalent to finding a sub-game perfect Nash equilibrium (SPNE).[5]

To avoid confusion, we emphasize that the training set is where the minibatch is sampled from and perturbed, the validation set is clean and public, and the test set is clean and unknown to both players. In our experiments, we evaluate their performance using the test accuracy. To allow both players to evaluate their utilities, we assume that a public validation set $\mathcal{D}_{\text{val}}$ is available to both players. However, in practice, the attacker can evaluate its utility even if $\mathcal{D}_{\text{val}}$ is unknown, i.e., simply by replacing $\mathcal{D}_{\text{val}}$ with the known $\mathcal{D}_t$ which is sampled from the same data distribution [6].[6] In contrast, the known validation set $\mathcal{D}_{\text{val}}$ is required by the defender since it does not have access to the clean minibatches. This is in the same spirit as the works on adversarial defense with data sanitization [17–19] which assume that the defender has clean data for training the de-noising models. These defense methods reform the perturbed inputs towards the distribution of clean inputs using an auto-encoder [18], a *variational auto-encoder* (VAE) [17], or a *generative adversarial network* (GAN) [19]. In contrast, instead of naively reforming the perturbed inputs using a de-noising model, our defender aims at directly reducing the validation loss in a principled way. Empirical evaluations show that our defender can potentially outperform perfect reconstruction (Sec. 6.3).

3.1. Connections with adversarial training

In adversarial training, the attacker is playing against a level-0/null-strategy defender under our R2T2 framework. In general, most adversarial training methods [4,20,38,15,39,21,22] share the following utility function of the attacker to be maximized:

$$\textbf{Attacker:} \quad \max_{\delta_t(x_t):\|\delta_t(x_t)\|_\infty \le 1} \left[\mathcal{L}_{\theta_t}(x'_t) - \mathcal{L}_{\theta_t}(x_t)\right]$$

whose utility function is different from $\mathcal{U}_t(x''_t)$ (5) in our framework. A key difference with adversarial training is that their utility function neglects the effect of training (i.e., no θ_{t+1} involved). This approximation is justified in their setting since test-time attacks, which they are defending against, occur only on the trained "static model". Therefore, in every base game of adversarial training, the attacker simulates such a test-time attack to the particular model instance in that base game.

4. Recursive reasoning-based training-time adversarial ML (R2T2)

This section introduces the R2T2 framework which makes use of the *level K* (LK) model to solve the game shown in (5) through recursive reasoning. In this framework, a level-0 player plays a null strategy and a level-$k \ge 1$ player chooses its strategy by best-responding to the level-$(k-1)$ opponent. We associate the cognitive limit and hence the reasoning level of the attacker or defender with the highest order of gradient that can be computed by the player:

Assumption 1 *(Cognitive limit). A level-k player can calculate and utilize at most the k-th-order gradients of the target ML model when deriving its strategy (i.e., the player can only calculate mixed partial $\nabla_\theta^a \nabla_x^b \mathcal{L}_\theta(x)$ where $a, b \le k$), and assumes all higher-than-k-th-order gradients ($a > k$ or $b > k$) to be* $\mathbf{0}$.

Under Assumption 1, we can formally define the LK model for the R2T2 framework and illustrate it in Fig. 2:

Definition 1 *(LK). A level-$k \ge 1$ player best-responds to the level-$(k-1)$ opponent by solving* (5) *using the k'-th-order gradients of the target ML model for all $k' \le k$.*

This notion of cognitive limit preserves the inherent relationship in the LK model between the required computational resources and performance: As k increases, the performance of the level-k strategy is likely to improve. However, this requires greater computational resources to derive the level-k strategy since the cost of computing the k-th-order gradient grows exponentially in k. As an analogy, second-order optimization techniques usually have a better convergence guarantee than first-order ones but require more computations. Such a relationship will be explored in our experiments in Sec. 6.3. Note that although a level-k player best-responds to the level-$(k-1)$ opponent's strategy, the player can perform effectively (a) even if the opponent does not reason at level $k-1$ but a lower level or (b) when the opponent does not follow our recursive reasoning framework and instead adopts some existing attack/defense baselines, as demonstrated in our experiments (Sec. 6). In the subsections to follow, let the level-k attack (1) and defense (2) strategies be denoted by $\delta_t^k(x_t)$ and $\sigma_t^k(x'_t)$, respectively. Similarly, let their optimal strategies be denoted by $\delta_t^{k^*}(x_t)$ and $\sigma_t^{k^*}(x'_t)$, respectively.

[5] Note that in our framework, the calculation of the optimal strategy is under the recursive reasoning formalism. For example, a level-2 attacker's optimal strategy will form a SPNE with a level-1 defender's optimal strategy subject to the constraint on the reasoning level of each player.

[6] Such details and experimental results are in Appendix A.2.4.

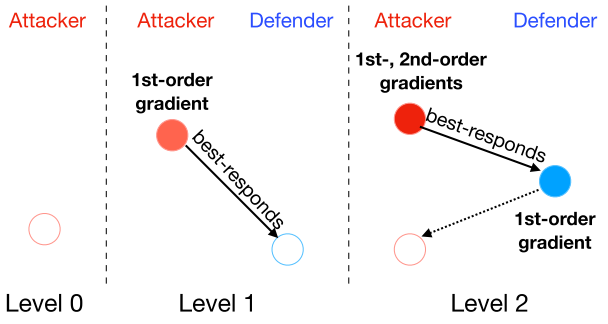


Fig. 2. Illustration of how a level-$k = 0, 1, 2$ attack strategy is computed under the LK model for R2T2 framework.

4.1. R2T2 level-0 (null) strategies

A level-0 player asserts that the model parameters are not updated (i.e., $\theta_{t+1} = \theta_t$) since all gradients are $\mathbf{0}$ due to Assumption 1 (i.e., $\nabla_\theta \mathcal{L}_{\theta_t}(x'_t) = \mathbf{0}$). So, both players think their utilities are *constants*. Without loss of generality, we assume that both players play the null strategies: $\delta_t^{0^*}(x_t) \triangleq \mathbf{0}$ and $\sigma_t^{0^*}(x'_t) \triangleq \mathbf{0}$ which correspond to no attack/defense. In fact, the level-0 strategies can be arbitrary and do not affect the functional form of the higher level-$k \geq 1$ strategies under our R2T2 framework, as proven in Appendix B.1.

4.2. R2T2 level-1 strategies

Theorem 1 *(R2T2 Level-1 strategies). The optimal level-1 attack and defense strategies are*[7]

$$\delta_t^{1^*}(x_t) = \vec{\mathbf{e}}\left(\nabla_x \mathcal{U}_t(x_t)\right), \quad \sigma_t^{1^*}(x'_t) = -\vec{\mathbf{e}}\left(\nabla_x \mathcal{U}_t(x'_t)\right) \tag{6}$$

where $\vec{\mathbf{e}}(\cdot)$ *outputs the unit vector of its input and*

$$\begin{aligned} \nabla_x \mathcal{U}_t(x) &= -\eta\, \mathbf{B}_{\theta_t}^x \left[\textstyle\sum_{z \in \mathcal{D}_{val}^X} \nabla_\theta \mathcal{L}_{\theta_t}(z)\right], \\ \mathbf{B}_{\theta_t}^x &\triangleq \left[\nabla_x \nabla_\theta \mathcal{L}_{\theta_t}(x)\right]^\top. \end{aligned} \tag{7}$$

Its proof is in Appendix B.1. Theorem 1 suggests that the optimal level-1 strategies follow from linearizing the utility functions, as implied by the linearized loss function under Assumption 1. The *deep confuse* (DC) strategy [6] can be interpreted as a variant of our level-1 attack strategy. Specifically, it trains an auto-encoder to learn an averaged $\delta_t^{1^*}$ over $t = 0, \ldots, T-1$ (i.e., over the trajectory of model training), which makes their attack strategy independent of t.[8] An advantage of their method is that their strategy can be computed using the auto-encoder without explicit gradient calculation. Its downside is that every trial (500 trials in total) of the auto-encoder training requires a full trajectory of model training which includes T iterations of gradient calculation and is hence time-consuming.

Some well-known adversarial training methods can be interpreted as level-1 attack strategies: (a) *fast gradient sign method* (FGSM) perturbing inputs with first-order gradient-based attacks [4], and (b) adversarial transformation network learning $\delta_t^{1^*}$ with first-order information [40].

4.3. R2T2 level-2 strategies

Level-2 players are capable of computing the Hessian $H_{\theta_t} \triangleq \sum_{z \in \mathcal{D}_{\text{val}}^X} \nabla_\theta^2 \mathcal{L}_{\theta_t}(z)$ w.r.t. θ.

Theorem 2 *(R2T2 Level-2 strategies). The optimal level-2 attack and defense strategies as* ϵ *approaches* 0 *are*

$$\begin{aligned} \lim_{\epsilon \to 0} \delta_t^{2^*}(x_t) &= -\vec{\mathbf{e}}\left(\mathbf{B}_{\theta_t}^{x_t}\left[\sum_{z \in \mathcal{D}_{val}^X} \nabla_\theta \mathcal{L}_{\theta_t}(z) - \eta H_{\theta_t} \nabla_\theta \mathcal{L}_{\theta_t}(x_t)\right]\right), \\ \lim_{\epsilon \to 0} \sigma_t^{2^*}(x'_t) &= \vec{\mathbf{e}}\left(\mathbf{B}_{\theta_t}^{x'_t}\left[\sum_{z \in \mathcal{D}_{val}^X} \nabla_\theta \mathcal{L}_{\theta_t}(z) - \eta H_{\theta_t} \nabla_\theta \mathcal{L}_{\theta_t}(x'_t)\right]\right). \end{aligned}$$

[7] Unless otherwise stated, the gradient of a scalar and the vectors are all column vectors.

[8] DC also differs from R2T2 by assuming access to the entire training set and not a known validation set $\mathcal{D}_{\text{val}}$.

Its proof (Appendix B.2) utilizes the idea that since the search space has a bounded L_2 norm and a level-2 player assumes the k-th-order gradients to be $\mathbf{0}$ for all $k \geq 3$ (Assumption 1), finding the optimal level-2 strategy of a player can be framed as a constrained quadratic programming problem and is tractable as ϵ approaches 0.

Interestingly, the Influence Function-based dataset poisoning attack [25] can be interpreted as a special case of our level-2 attack when the Hessian H_{θ_t} is positive definite, as discussed below.

Optimal Contraction (OC) & Connections with Influence Function Attacks (IF). A further insight on our level-2 strategy can be drawn when the Hessian H_{θ_t} is positive definite. In this case, *optimal defender's parameters* $\theta^\star \triangleq \arg\min_\theta \sum_{z\in\mathcal{D}_{\text{val}}^X} \mathcal{L}_\theta(z)$ exist s.t. the defender achieves minimum validation loss since now, $\sum_{z\in\mathcal{D}_{\text{val}}^X} \mathcal{L}_\theta(z)$ is a convex polynomial of θ.[9] So, instead of optimizing the base game utility (5), the defender can choose to directly minimize $\|\theta_{t+1} - \theta^\star\|$ in each iteration t to make the parameters of the target ML model move close to $\theta^\star$. In the ideal case where $\theta^\star$ is reached (i.e., $\|\theta_T - \theta^\star\| = 0$), the defender achieves maximum extensive-form game utility (4).

In reality, the defender may not want to naively set $\theta_{t+1} = \theta^\star$ because in such a case, the model cannot utilize any information from the training data, which is an undesirable behavior. So, a realistic assumption is that the defender would only utilize the optimal parameters as a soft constraint by restricting θ_{t+1} from getting too far away (from the optimal parameters for the validation set). In this sense, the defender is said to perform the *optimal contraction defense* strategy. Conversely, the attacker who knows the existence of the optimal parameters would try to deviate from it. This gives rise to the *optimal contraction attack* strategy; we have abused the use of "optimal contraction" here for simplicity as the attacker would in fact try its best to undo the contraction, as revealed below:

Corollary 1 *(Optimal contraction (OC) attack and defense). Suppose that H_{θ_t} is positive definite. The OC strategy $\delta_t^{\text{OC}}(x_t) \triangleq \arg\max_{\delta_t(x_t):\|\delta_t(x_t)\|\leq 1} \|\theta_{t+1} - \theta^\star\|$ of a level-2 attacker and the OC strategy $\sigma_t^{\text{OC}}(x_t) \triangleq \arg\min_{\sigma_t(x_t):\|\sigma_t(x_t)\|\leq 1} \|\theta_{t+1} - \theta^\star\|$ of a level-2 defender as ϵ approaches 0 are, respectively,*

$$\lim_{\epsilon\to 0} \delta_t^{\text{OC}}(x_t) = -\vec{\mathbf{e}}\left(\mathbf{B}_{\theta_t}^{x_t}\left[H_{\theta_t}^{-1}\sum_{z\in\mathcal{D}_{val}^X} \nabla_\theta \mathcal{L}_{\theta_t}(z) - \eta\nabla_\theta \mathcal{L}_{\theta_t}(x_t)\right]\right),$$

$$\lim_{\epsilon\to 0} \sigma_t^{\text{OC}}(x_t) = \vec{\mathbf{e}}\left(\mathbf{B}_{\theta_t}^{x_t}\left[H_{\theta_t}^{-1}\sum_{z\in\mathcal{D}_{val}^X} \nabla_\theta \mathcal{L}_{\theta_t}(z) - \eta\nabla_\theta \mathcal{L}_{\theta_t}(x_t)\right]\right).$$

Its proof is in Appendix B.2. Interestingly, the IF strategy [25] also assumes a positive definite Hessian and adopts a similar attack strategy as OC:

$$\delta_t^{\text{IF}}(x_t) \triangleq -\vec{\mathbf{e}}\left(\mathbf{B}_{\theta_t}^{x_t}\left[H_{\theta_t}^{-1}\textstyle\sum_{z\in\mathcal{D}_{\text{val}}^X} \nabla_\theta \mathcal{L}_{\theta_t}(z)\right]\right). \tag{8}$$

Proven to be suitable for data poisoning, IF can also be used directly for training-time attacks. By comparing (8) with Corollary 1, IF can be viewed as an approximation of OC by ignoring the $\eta\mathbf{B}_{\theta_t}^{x_t}\nabla_\theta \mathcal{L}_{\theta_t}(x_t) = (\eta/2)\nabla_x\|\nabla_\theta \mathcal{L}_{\theta_t}(x_t)\|^2$ term. Mathematically, this term tends to reform the input to increase the norm of model gradients $\|\nabla_\theta \mathcal{L}_{\theta_t}(x_t)\|^2$. This is indeed observed in our experiments and we discover that the addition of this term can potentially further intervene in the training, thus resulting in slightly better attacks.

However, OC and IF are restrictive in two aspects: (a) They require the existence of $\theta^\star$ and access to its tractable expression, which is only guaranteed with a level-2 assumption and a positive definite Hessian. (b) Even with access to a tractable expression of $\theta^\star$, OC and IF may not stay away from $\theta^\star$ in the long run because the defender (i.e., if it reasons at level $k \geq 1$) or the model training itself may reperform the contraction, thus potentially decreasing $\|\theta_{t+1} - \theta^\star\|$ over t even though the attacker tries to maximize it. So, our level-2 attack strategy in Theorem 2 is more general since it does not require the existence of (or any knowledge about) $\theta^\star$. An extensive empirical comparison in Sec. 6.2 shows that our level-2 attack strategy is indeed more effective. During implementation, we follow the work of [25] and approximate the Hessian via conjugate gradients. A few adversarial training methods [38,39] can be classified as second-order attack strategies (albeit based on a different utility function (Sec. 3.1) and without accounting for defense) and approximate the calculation of the Hessian by power iterations.

4.4. R2T2 level-$k \geq 2$ strategies

Deriving the optimal level-$k \geq 2$ strategies requires finding the optimum of $k \geq 2$-th-order polynomials with constraints, which is generally intractable. Calculating the optimal higher-level strategy tractably may not be of practical interest since it involves the computation of higher-order gradients, which is time-consuming for modern deep learning models due to huge number of parameters. So, we resort to approximation methods to solve for level-$k \geq 2$ strategies. Some previous works can

[9] Due to cognitive limit, a level-2 player visualizes the loss function as a second-order polynomial.

be viewed as heuristics to approximate higher-level test-time attacks by *projected gradient descent* (PGD) [20,15] that do not account for defenses.

In this subsection, we propose different training-time PGDs to approximate the optimal level-$k \geq 2$ attack and defense strategies under our R2T2 framework. We show that the convergence of our PGDs can be guaranteed under convexity and other mild conditions and the number of PGD steps required to approximate the optimal level-$k \geq 2$ strategy to a constant accuracy is exponential in k.

Remark 1 *(Attacker's* nested PGD *(NPGD)). The following NPGD approximates the optimal level-$k \geq 2$ attack strategy $\delta_t^{k^*}$ at input x_t with $\delta_{[i]}$ and starts from PGD step $i = 0$ with $\delta_{[0]} = \sigma_{[0]} = \mathbf{0}$:*

$$\begin{aligned} \delta_{[i+1]} &= \operatorname{Proj}\left(\delta_{[i]} + (\Gamma_k/\epsilon)\nabla_x \mathcal{U}_t\left(x'_{[i]} + \epsilon\, \sigma_{[i]}\right)\right), \\ \sigma_{[i+1]} &= \operatorname{Proj}\left(\sigma_{[i]} - (\Gamma_{k-1}/\epsilon)\nabla_x \mathcal{U}_t(x''_{[i+1]})\right) \end{aligned} \tag{9}$$

where $x'_{[i]} \triangleq x_t + \epsilon\, \delta_{[i]}$, $x''_{[i+1]} \triangleq x'_{[i+1]} + \epsilon\, \sigma_{[i]}$, Γ_k defines the step size to approximate $\delta_t^{k^*}$, $\operatorname{Proj}(x) \triangleq \vec{\mathbf{e}}(x) \times \min(\|x\|, 1)$ is a projection to ensure $\|\delta_{[i+1]}\|, \|\sigma_{[i+1]}\| \leq 1$,[10] and $\nabla_x \mathcal{U}_t(x)$ is given in (7).

Remark 2 *(Defender's PGD). The following PGD approximates the optimal level-$k \geq 2$ defense strategy $\sigma_t^{k^*}$ at input x'_t with $\sigma_{[i]}$ and starts from PGD step $i = 0$ with $\sigma_{[0]} = \mathbf{0}$:*

$$\sigma_{[i+1]} = \operatorname{Proj}\left(\sigma_{[i]} - (\Gamma_k/\epsilon)\nabla_x \mathcal{U}_t(x''_{[i]})\right) \tag{10}$$

where $x''_{[i]} \triangleq x'_t + \epsilon\, \sigma_{[i]}$.

By comparing (9) and (10), it can be observed that the attacker follows a more complicated PGD because in each iteration t, the attacker adds its perturbations first and hence does not observe how the defense strategy (which adapts to the attacks) transforms the perturbed inputs in the same iteration. This then requires the attacker to additionally know the level-$(k-1)$ defense strategy $\sigma_t^{(k-1)^*}$ to perform PGD, which cannot be computed analytically in practice. So, it utilizes our proposed NPGD which approximates $\sigma_t^{(k-1)^*}$ with $\sigma_{[i]}$ by recursively reasoning about the defender's behavior (i.e., second PGD step of (9)).

Theorem 3 *(Convergence of defender's PGD). Suppose that $\mathcal{U}_t(x)$ is a convex function of x. Then, there exists a constant M s.t. by setting the step size $\Gamma_k \triangleq 1/[M^2k^2(1+2M\epsilon)^{k-2}]$, the defender's PGD (10) has the following convergence guarantee in PGD step i:*

$$\left|\mathcal{U}_t\left(x'_t + \epsilon\, \sigma_{[i]}\right) - \mathcal{U}_t\left(x'_t + \epsilon\, \sigma_t^{k^*}(x'_t)\right)\right| \leq 2\epsilon^2/(\Gamma_k i)\ .$$

Furthermore, if $\mathcal{U}_t(x)$ is μ-strongly convex, then

$$\left\|\sigma_{[i]} - \sigma_t^{k^*}(x'_t)\right\| \leq \exp(-\mu\Gamma_k i)\ .$$

Its proof is in Appendix B.3 and builds upon that for the convergence guarantee of PGD [41]. If the attacker's approximation of $\sigma_t^{(k-1)^*}$ is accurate, then the attacker's NPGD enjoys a similar guarantee as the defender's PGD (Appendix B.3). A key ingredient for deriving Theorem 3 is Assumption 1 which turns out to constrain the smoothness of the loss function at a particular reasoning level. Note that the convexity assumptions are not guaranteed in practice. However, they are only adopted for the purpose of the theoretical analysis and thus not strictly required in practice for our PGD to deliver an effective performance.

Setting the step size and number of PGD steps. The convergence guarantee indicates that a desired accuracy can be achieved with a step size of $\Gamma_k \triangleq 1/[M^2k^2(1+2M\epsilon)^{k-2}]$ in $i = \mathcal{O}(k^2(1+2M\epsilon)^k)$ PGD steps where M is a small constant that can be estimated heuristically (Appendix B.3.2). In our experiments, we thus approximate the optimal level-k attack and defense strategies, respectively, through (9) and (10) by heuristically estimating M and setting the number of PGD steps according to Appendix B.3.2. When $\Gamma_1 \triangleq \infty$, performing 1 step of the attacker's NPGD and defender's PGD exactly recovers the level-1 strategies in Theorem 1. Both the attacker's NPGD and the defender's PGD can be computed for all the inputs of the minibatch in parallel using modern deep learning libraries.

[10] For image inputs in our real-world experiments, we also ensure that the pixel values are within the valid range at all time.

Table 1
Unifying existing adversarial ML methods under our R2T2 framework which classifies them as attackers or defenders with different reasoning levels. Level-k (i.e., last row) means that the method is not restricted to a specific reasoning level. The methods marked with * can be easily adapted to the setting of training-time adversarial ML considered in our work here, while the methods marked with † explicitly consider adversarial defense.

Reasoning Level of Attacker	Reasoning Level of Defender	Adversarial ML Method	Perturbation Constraint	Original Paradigm
1	0	DC [6]*	L_∞	Training-time attack
	0	FGSM [4]	L_∞	Test-time attack
	0	ATN [40]	Any	Test-time attack
	k	defense-GAN [19]*†	Any	Test-time defense
2	0	IF [25]*	L_2	Data poisoning
	0	VAT [39], S-O [38], CRT [21]	L_2	Adversarial training
k	0	FGSM-PGD [20], FGSM-PGD [15]	L_∞	Adversarial training
	k	R2T2 (Ours)	L_2	Training-time attack/defense

4.5. Unifying existing adversarial ML methods under our R2T2 framework

Table 1 shows how our R2T2 framework can unify various existing adversarial ML methods and classify them as attackers or defenders with different reasoning levels, as discussed in the previous subsections.

We would like to clarify that our Assumption 1 states that a level-k player would *regard* the higher-order ($>k$) gradients to be zero. In this case, when a player is using (at most) the k-th order gradient, we characterize it as a level-k player, even if its k-th order gradient is being approximated (e.g., by multiple evaluations of the level $(k-1)$-th order gradient). This aligns with our fundamental idea/perspective discussed in the paragraph after Definition 1: The characterization of the reasoning level (hence, cognitive limit) is determined by the required computational resources. That is, computing level-k gradients (even if approximated) will inevitably incur a greater computational cost than computing level-$(k-1)$ gradients. As an example, in some of our experiments, a level-2 adversary uses conjugate gradient to approximate the Hessian, which is a realistic experimental setup since the CNN model has a large number of parameters. Alternatively, we can choose to compute the exact Hessian if the available computational resources permit. However, note that both the Hessian approximation via conjugate gradient or the exact Hessian calculation incur a greater computational cost than computing the 1-st order gradient.

4.6. Generalization to black-box attack scenario

We examine a different scenario where the model parameters and the model architecture are not known to the attacker. Attacks under such a scenario are known as *black-box attacks*. To realize black-box attacks, we assume the attacker possesses the entire training dataset.[11] This is a common assumption so that the attacker can train a local model to reason about the training process [6].

Before the game starts, the attacker trains a local model $\theta_t^\star$ for $t = 0, \ldots, T'$, and meanwhile the attacker performs our level-k white-box attacks on the local model during training. The whole process lasts until the attack performance (i.e., validation loss) on the local model converges and the final perturbation $\delta_{T'}^{k*}(x)$ on every input x in the training dataset is recorded down. During the game, when a minibatch (i.e., selected from the same dataset that the attacker possesses) is being transmitted to the central server, the attacker identifies each input x_t in the minibatch and adds a corresponding perturbation $\delta_{T'}^{k*}(x_t)$ based on its record; if the training uses a different training dataset from the dataset that the attacker possesses and x_t is not in the record, then the attacker can compute the perturbation using our white-box attack strategy by setting $\theta_t = \theta_{T'}^\star$. On the other hand, since the defender is on the central server, we assume the defender always knows the true model parameter. Thus the defender always conducts white-box defenses.

5. A synthetic experiment

We first use a synthetic experiment to show that by reasoning at a higher level, a player is able to find a better strategy by behaving more intelligently. We consider the game of a linear regression task with the ground truth model $y(x) = \theta^{*\top} x$ and squared error loss function $\mathcal{L}_{\theta_t}(x) = \|\theta_t^\top x - \theta^{*\top} x\|^2$. In iteration t, the attacker intends to increase $\|\theta_{t+1} - \theta^*\|$ to prevent

[11] The training dataset that the attacker possesses does not need to be the same as the training data used for actual model training, but is assumed to be from the same underlying distribution.

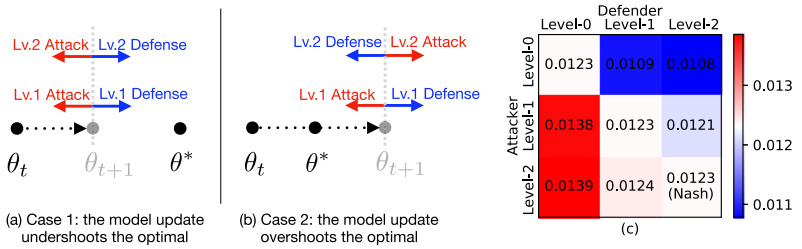


Fig. 3. Behaviors of level-1 and level-2 players when the original updated model parameters θ_{t+1} (i.e., not subject to attack/defense) (a) undershoots and (b) overshoots the optimal parameters θ^*. (c) shows mean test loss after 30 training iterations.

the model from learning the optimal parameters θ^*; meanwhile, the defender attempts to decrease it. In this experiment, the optimal level-1 and level-2 strategies can be computed tractably.[12] The results reveal that when the original updated model parameters θ_{t+1} (i.e., not subject to attack/defense) undershoots θ^* (Fig. 3a), both level-1 and level-2 attackers (defenders) behave correctly since both attackers (defenders) learn to move θ_{t+1} away from (towards) θ^*. However, in the more challenging scenario where θ_{t+1} overshoots θ^* (Fig. 3b), only the level-2 attacker (defender) is able to increase (decrease) the distance between θ_{t+1} and θ^*, while the level-1 attacker (defender) fails, as explained in Appendix A.2.1. Interestingly, in this experiment, Nash equilibrium is attained from level 2 onwards, i.e., every optimal level-$k \geq 2$ strategy is the Nash equilibrium strategy (Appendix A.2.1). Fig. 3c reveals the benefit of reasoning at a higher level. Fig. 3c also shows that when the opponent's level is fixed at 0, reasoning at a higher level (2 instead of 1) still brings benefit, although the gain diminishes (i.e., proceeding from level 1 to 2 offers less marginal benefit than from progressing level 0 to 1).

6. Real-world experiments and discussion

We perform real-world experiments on MNIST, CIFAR-10/100, and a sub-sampled 2-class ImageNet dataset (2000 images of size $256 \times 256 \times 3$ each) of *owl* and *jellyfish*. All inputs are images whose pixel values lie within $[0, 1]$. For MNIST, CIFAR-10, and ImageNet, we set ϵ to be 3, 2, 4[13]; minibatch size to be 500, 400, 100; target ML model to be *convolutional neural networks* (CNN) with around 0.1M, 1M, and 1M parameters, respectively. We split 5000 images from the training data of MNIST and CIFAR, and 250 images from the ImageNet to serve as the validation set for each task. For other methods under comparison, we normalize their perturbations (transformations) to enforce the bounded L_2 norm constraint. Appendix A.1 gives more details on the network architectures, hyperparameter settings, among others.

6.1. R2T2 level-1 strategies

R2T2 Level-1 Attack. We compare the performance of our level-1 attack against that of other attack methods including test-time attack/adversarial training baselines of FGSM-PGD [15], CW [14], and YOPO [16], and the training-time attack baselines of DC [6] and EvoShift [42]. EvoShift is a non-gradient-based attack which uses an evolutionary algorithm to optimize the pixel patterns (i.e., one for each class to be added to the training images), so the targeted ML model may learn to classify purely based on the injected pixel patterns. For FGSM-PGD, we perform 5 PGD steps in total. For CW, we follow its original implementation (with L_2 constraint and hinge loss on logits). For YOPO, we implement YOPO-5-3. For EvoShift, we adopt the original setting of [42] to modify 5 pixels per image and perform the evolution of its perturbation jointly with the model training.

The baselines (except DC and EvoShift) are not designed for training-time attack and thus do not attack well even if against a null-strategy defender (Table 2). EvoShift does not perform as well as that in the original paper [42], likely due to the small ϵ_{atk} value or every pixel value falling within a valid range between 0 to 1 (i.e., $0 \sim 255$ in the unnormalized image). We have found that if the valid pixel range is not enforced, then EvoShift (against no defense) can substantially reduce the test accuracy to be close to a random guess after ϵ_{atk} is increased to a larger magnitude. Our level-1 attack outperforms the other attacks and decreases the test accuracy considerably in most cases (Table 2 and Fig. 4a), except when against our level-1 defender on MNIST. Results on CIFAR-10 (Table 2) show that our black-box level-1 attacks (Sec. 4.6) can potentially outperform our white-box attacks, which agrees with recent discoveries on test-time attacks [22] that black-box attacks can sometimes be more effective. In Fig. 4a, note that DC achieves better attack performance in the early iterations ($<$50 epochs) because its attacker is pre-trained using the entire training dataset before the game starts. Figs. 4b and 4c show that ϵ_{atk} should not be too small for the attacker to deliver effective attacks and a level-1 defense will require ϵ_{atk} to be increased for the attacker to achieve similar attack effectiveness.

[12] More details, including the tractable expressions of the optimal attack/defense strategies, are provided in Appendix A.2.1.

[13] An image perturbation/transformation scaled by $\epsilon = 3, 2, 4$ for MNIST, CIFAR, and Imagenet has an equivalent bounded L_2 norm as that under the bounded L_∞ norm constraint scaled by $\varepsilon = 0.107, 0.036, 0.009$, respectively.

Table 2
Mean test accuracy (%) ± 1 standard deviation (i.e., over 5 runs) for attacks against level-0 and level-1 defenders.

Attacker \ Defender (Ours)	MNIST		CIFAR-10		CIFAR-100	
	Level-0	Level-1	Level-0	Level-1	Level-0	Level-1
No attack	98.8 ± 0.5	97.2 ± 0.8	74.8 ± 3.5	71.4 ± 2.2	55.4 ± 1.3	56.4 ± 1.9
FGSM-PGD [15]	80.5 ± 2.2	**91.3 ± 4.5**	45.6 ± 1.7	57.4 ± 2.6	40.5 ± 2.9	44.6 ± 3.8
CW [14]	75.4 ± 4.9	92.8 ± 3.2	50.7 ± 2.7	63.4 ± 3.7	38.4 ± 2.5	43.8 ± 3.6
YOPO [16]	87.5 ± 2.3	94.0 ± 3.8	45.2 ± 4.3	57.0 ± 2.5	39.6 ± 2.5	42.9 ± 4.9
DC [6]	28.5 ± 6.4	96.1 ± 1.9	24.7 ± 5.6	62.5 ± 5.3	19.6 ± 6.6	43.1 ± 3.9
EvoShift [42]	50.1 ± 3.2	95.9 ± 1.1	23.1 ± 5.4	67.3 ± 6.0	18.6 ± 2.6	48.3 ± 5.9
Level-1 (Ours)	**9.5 ± 2.9**	94.8 ± 2.8	17.1 ± 3.2	**55.3 ± 4.0**	**7.6 ± 1.5**	**42.8 ± 5.2**
Level-1 *black-box* (Ours)	11.7±2.2	95.5±1.7	**15.8±2.7**	60.9±5.9	7.9±1.9	44.2 ± 3.1

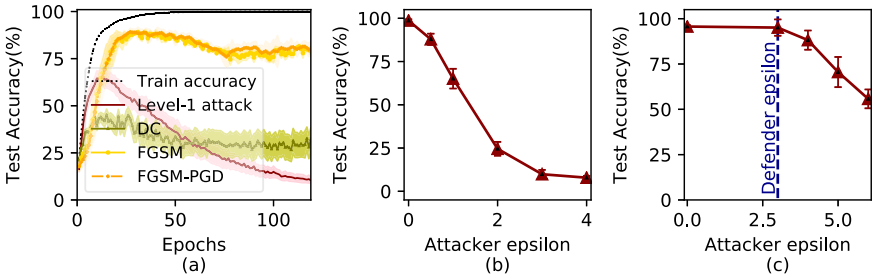


Fig. 4. (a) Learning curves for various attacks on MNIST against level-0 defender, and test accuracy for our level-1 attack on MNIST with varying ϵ_{atk} against (b) level-0 and (c) level-1 defenders. *Lower* accuracy reflects *better* attack. Error bars indicate standard deviations over 5 runs.

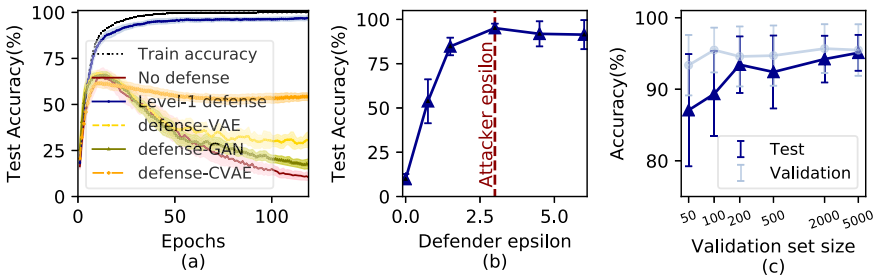


Fig. 5. (a) Learning curves for various defenses on MNIST against level-1 attacker, and test accuracy for our level-1 defense on MNIST with varying (b) ϵ_{def} and (c) size of validation set known to the defender. *Higher* accuracy reflects *better* defense performance. Error bars indicate standard deviations over 5 runs.

R2T2 Level-1 Defense. Under level-1 attack, we compare the performance of our level-1 defense with the other defenses[14] such as defense-VAE [17], defense-GAN [19], and defense-CVAE where we replace the VAE by a *conditional VAE* (CVAE). The generative models of those baselines (implemented with CNN) are trained on the known validation set. The GAN is carefully tuned to avoid mode collapse.

The results are shown in Fig. 5a. Our level-1 defense strategy is proven to be effective and largely outperforms existing methods, and it is also effective when defending against the other attack methods (Table 2). Fig. 5b shows that ϵ_{def} should be in a scale similar to ϵ_{atk} to achieve good defense performance, and further increasing ϵ_{def} may result in slight overfitting to the validation set. Fig. 5c shows that when a much smaller validation set is used, overfitting to the validation set can also be observed.

In addition, we further examine the defense setting where the defender is allowed to directly train on the validation set. In contrast, the above-mentioned defense methods are only allowed to utilize the validation set to bootstrap a defense model. We assume training on the validation set to happen constantly (i.e., every epoch) during training on the perturbed training data. We implement the state-of-the-art baseline of FixAug [43] in the RobustBench leaderboard for "CIFAR-10 with L_2 norm of 0.5".[15] FixAug comprises a generative denoising model (i.e., similar to defense-GAN or defense-CVAE introduced

[14] Results on non-gradient-based defense strategies are reported in Appendix A.2.3.
[15] The RobustBench leaderboard is at https://robustbench.github.io.

Table 3
Mean test accuracy (%) on MNIST and CIFAR-10. The data augmentation method of CutMix is applied either on the training set or on both training and validation sets. For both MNIST and CIFAR-10, the size of the validation set is 5000.

Defender \ Attacker	MNIST		CIFAR-10	
	Level-1	Level-4	Level-1	Level-4
DDPM	27.9	19.9	19.8	14.4
DDPM + CutMix (training set)	33.7	25.4	20.8	15.3
DDPM + CutMix (training & **validation sets**)	58.6	46.4	37.6	31.0
Level-1	93.6	92.1	55.3	48.6
Level-1 + CutMix (training set)	94.2	92.3	55.4	51.0
Level-1 + CutMix (training & **validation sets**)	**95.3**	**92.9**	**56.7**	**54.5**

earlier) and a data augmentation method. When applied to training-time defense, we have adopted the best reported setting which is a combination of the defense model of DDPM [44] and the data augmentation method of CutMix [45]. The denoising model of DDPM is learned with the validation set and used as the defense strategy to denoise the perturbed training data.[16] To migrate to our setting, we set the hyperparameters carefully (i.e., $T = 500$ and β_t increases linearly from $1e^{-4}$ to $\epsilon^2_{\text{def}}/\dim(x)$) according to the fact that the perturbation in our training data is known to be L_2-bounded. The reverse process (i.e., generative model) is a wide residual network which is the same backbone architecture used in the original paper [44] but smaller in size ($\sim 1/10$ of the number of parameters) because we are only training on the validation set of size 5000. The other hyperparameter settings are the same as that in [44].

CutMix is the only data augmentation method used when the ML model is trained on the validation set. We have additionally compared the performance of the above baseline with that of a combination of our level-1 defense strategy and CutMix. CutMix is applied either only on the training set or on both the training and validation sets when we additionally train on the validation set. The combination ratio is heuristically set by following that in the original paper [45]. The results are presented in Table 3.

The ablation studies in Table 3 show that the state-of-the-art data augmentation method of CutMix for improving test-time robustness only yields limited performance improvement when only applied on the perturbed training data. This observation is in accordance with the performance of another state-of-the-art baseline of Crop Ensemble in Appendix A.2.3. The unsatisfactory performance improvement is probably due to the fact that the images are already perturbed and those transformations/augmentations are not likely to remove the effects of the malicious perturbation.

However, when the defender additionally trains on the augmented validation set, the defense performance is clearly improved. As an example, for MNIST, it can bring about over 20% of accuracy improvement for DDPM. This is possibly due to our validation set size being not negligible (i.e., around 1/10 of the training set). Note that the performance of DDPM against level-1 attacker on MNIST (i.e., 27.9%) is better than most generative-based defense models (see Fig. 5a) but not as good as our carefully designed defense-CVAE which is trained with conditional information, which is expected since the training of DDPM does not utilize the conditional information.

Table 3 also shows that our level-1 defense strategy outperforms DDPM, and the best defense performances are obtained via the combination of our level-1 defense and CutMix (on both training and validation sets). Moreover, these best performances are better than that of models trained solely on the validation set (with data augmentation), which achieve 91.2% and 51.8% accuracy for MNIST and CIFAR-10, respectively. This reveals the effectiveness of our defense strategy which can exploit the perturbed training data for achieving an improvement compared with trivially not using it at all.

6.2. Experiments on level-2 attacks: R2T2 level-2, OC, and IF

We examine three attacks discussed in Sec. 4.3: our R2T2 level-2 attack (Theorem 2), its variant called OC, and IF [25]. We first attack the training of a binary logistic regression classifier on a subsampled MNIST dataset of digits 1 vs. 7. This is a proof-of-concept experiment where the Hessian of the model is guaranteed to be positive definite. We then attack the training of a CNN (10-class) where the Hessian is not guaranteed to be positive definite.

It can be observed from Fig. 6 that IF and OC perform nearly identically for binary classification but differently for CNN. As analyzed in Sec. 4.3, IF ignores the "gradient increment" term $(\eta/2)\nabla_x \|\nabla_\theta \mathcal{L}_{\theta_t}(x)\|_2^2$ and thus results in a smaller gradient norm in the CNN experiment (Fig. 6c), which potentially explains why IF is outperformed by OC.

R2T2 level-2 attack outperforms OC and IF as we expected (Sec. 4.3). Though the performance improvement is already noticeable in binary classification, our R2T2 level-2 attack outperforms OC and IF by an even larger margin in CNN, potentially because our R2T2 level-2 attack is not subject to the constraint of a positive definite Hessian, as explained in Sec. 4.3.

[16] Note that we did not use DDPM to generate augmented data when the ML model is also trained on the validation set because we want a fair comparison between DDPM and our level-1 defense.

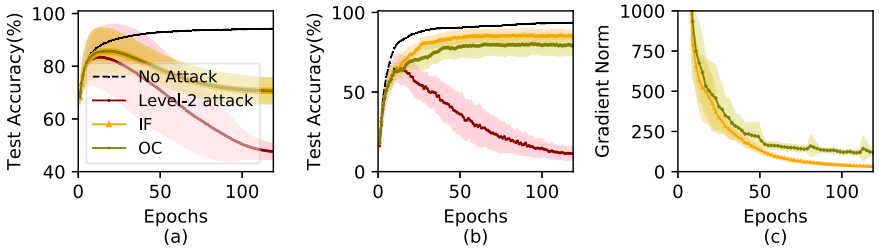


Fig. 6. Test accuracy of (a) binary classifier and (b) CNN classifier for 3 different level-2 attack methods, and (c) gradient norm $\sum_{x\in\mathcal{D}_t} \|\nabla_\theta \mathcal{L}_{\theta_t}(x)\|_2$ for CNN classifier. The error bars indicate standard deviations over 10 runs.

6.3. R2T2 level-k strategies

We investigate further using CIFAR-10 and ImageNet. We approximate the optimal level-k attack and defense strategies, respectively, through (9) and (10) by heuristically setting the number of PGD steps according to Appendix B.3.2. From Fig. 7a, inspection of a specific column or row reveals that reasoning at a higher level can still bring benefit even if against a fixed-level opponent. From Fig. 7b, it is more evident that in such a game, the marginal benefit of reasoning at a higher level quickly diminishes as a player's reasoning level goes beyond 1 (e.g., level-1 defender can improve over level 0 by 28% accuracy but level-4 defender can only improve over level 3 by 1%). Meanwhile, Fig. 7b also shows that the time needed to compute a higher-level strategy increases significantly with the level, which is in accordance with our theoretical analysis in Sec. 4.4. This implies that in practice, a player can only gain limited marginal benefit by executing significantly more steps of NPGD/PGD.

From the first row of Fig. 7c, we can observe that even if against a null-strategy attacker, a level-$k \geq 1$ defender improves test accuracy over a null-strategy defender, thus implying that our defense mechanism can potentially outperform perfect reconstruction (i.e., level-0 attacker vs. level-0 defender). This is indeed true as the defense alone can make the training process robust and improvement in the test accuracy can be due to the defender utilizing the clean validation set as an "augmentation" to the training data. Such a phenomenon is not always obvious on simple datasets like MNIST (Table 2) or CIFAR-10 (Fig. 7a), but is also observed in the more complex dataset CIFAR-100 (Table 2). The first row of Fig. 7d shows that the defense sometimes negates the attack's perturbation (in our visualization, green is the negation of purple), while the second row shows that the defense is sometimes more abstract to interpret. More visualizations are provided in Appendix A.2. The relationship between test accuracy and time needed to compute the strategies for ImageNet and MNIST, and a table of test accuracy for MNIST are also provided in Appendix A.2, both of which show similar trends to the above.

6.4. Generalization to targeted attacks

We also investigate the case when the attacker is *targeted*. The first scenario is the *conversion attack* where the attacker intends to misguide the classifier model into outputting a targeted label during test time. The attacker's utility function can be represented as $\mathcal{U}_t^\star$ and the base game becomes

$$\textbf{Attacker:} \quad \max_{\delta_t(x_t):\|\delta_t(x_t)\|\leq 1} \mathcal{U}_t^\star(x_t'') \,,$$

$$\textbf{Defender:} \quad \max_{\sigma_t(x_t'):\|\sigma_t(x_t')\|\leq 1} -\mathcal{U}_t(x_t'') \,.$$

where $\mathcal{U}_t^\star \triangleq \sum_{z\in\mathcal{D}_{\text{val}}^X} \left[\mathcal{L}_{\theta_{t+1}}^\star(z) - \mathcal{L}_{\theta_t}^\star(z)\right]$ and $\mathcal{L}_{\theta_t}^\star(z)$ is the loss incurred if the prediction of z is not the targeted label. Note that we assume that the defender is not targeted and just wants to improve the test performance, hence adopting the same utility function as (5).

For the results in Table 4, we evaluate the performance of the attack by the conversion rate, that is, how many test predictions (whose ground-truth is not the targeted label) are successfully turned into the targeted label. Interestingly, it can be observed that the conversion attack on MNIST is not very successful but is relatively successful on CIFAR-10, and the level-1 defender (even though it has a indiscriminate utility function) can drastically reduce the success rate of the conversion attack.

Another targeted attack scenario is the *evasion attack* where the attacker's goal is to guide the model into *not* outputting a correct targeted label during test time. So now, $\mathcal{L}_{\theta_t}^\star(z)$ is the loss incurred if the prediction of z, whose ground truth is the targeted label, is correct.

For the results in Table 5, we evaluate the performance of the attack by the evasion rate, that is, how many test predictions (whose ground truth is the targeted label) are successfully turned into another label. Again, it can be observed that the evasion attack on CIFAR-10 is much more successful than on MNIST, possibly because it is easier to train well with MNIST to yield a more robust model against such attacks, and the level-1 defender (even though it has a indiscriminate utility function) can largely reduce the success rate of the evasion attack.

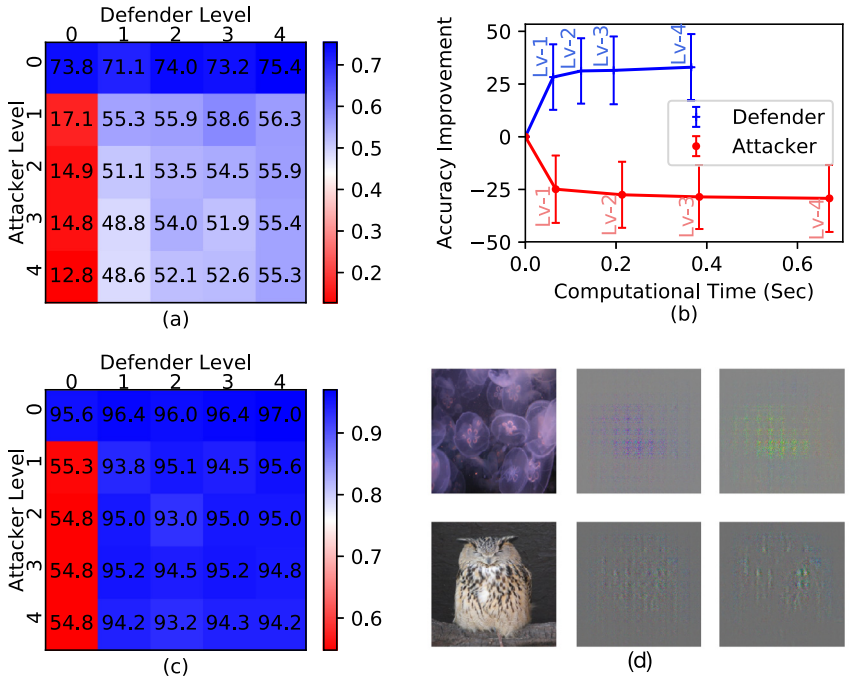


Fig. 7. (a) Mean test accuracy (%) on CIFAR-10. (b) Graph of mean accuracy improvement (over null strategy) vs. time needed to compute the strategies given a minibatch of CIFAR-10. (c) Mean test accuracy for 2-class ImageNet. (d) From left to right are visualizations of the original sampled images from ImageNet, perturbations from level-1 attacks, and transformations from level-1 defenses, respectively (color rescaled for viewing).

Table 4
Mean attack conversion rate (%) ± 1 standard deviation (i.e., over 5 runs) for targeted conversion attacks against level-0 and level-1 defenders. The higher the better.

	MNIST		CIFAR-10	
Attacker \ Defender	Level-0	Level-1	Level-0	Level-1
Level-1	23.7±9.6	2.8±2.3	78.1±3.0	15.7±9.7

Table 5
Mean attack evasion rate (%) ± 1 standard deviation (i.e., over 5 runs) for targeted evasion attacks against level-0 and level-1 defenders. The higher the better.

	MNIST		CIFAR-10	
Attacker \ Defender	Level-0	Level-1	Level-0	Level-1
Level-1	37.5±12.0	14.4±10.3	98.1±0.8	20.4±13.4

7. Conclusion

This paper describes the R2T2 framework which allows us to derive competitive level-k attack and defense strategies for the game of training-time adversarial ML. We empirically demonstrate that such strategies can achieve state-of-the-art performances on various benchmark image datasets. For our future work, we will investigate non-myopic approaches to solve this extensive-form game by considering a few steps lookahead.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Yizhou Chen reports financial support, administrative support, and equipment, drugs, or supplies were provided by National University of Singapore.

Data availability

No data was used for the research described in the article.

Acknowledgements

This research/project is supported by the National Research Foundation Singapore and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-RP-2020-018).

Appendix A. Supplementary information and more experimental results

A.1. Additional information for experiments

A.1.1. Stabilizing training with momentum

As discovered by [6], a naive implementation of training-time attacks may keep the model training from convergence: In two consecutive iterations, the perturbations added by the attacker can be significantly different, which may cause dramatic differences in the modified data distributions, thus rendering the model training difficult. This is undesirable since the attacker's goal is to fool the target ML model into performing badly during test time while ensuring its convergence during training.

In order to tackle this instability during training (i.e., sometimes also observed in our experiments), we adopt a *momentum-based* variant of the attack strategy that keeps track of the attacker's own modification history of an input x_t as $\delta^H_{t-1}(x_t)$. The history starts from $\delta^H_0 = 0$ and is updated when the attacker performs attacks; if an input is not in $\mathcal{D}^X_t$, then the history of this input will not be updated in iteration t. Then, the momentum-based variant of the optimal level-k attack strategy is

$$\overline{\delta^{k^*}_t}(x_t) = \text{Proj}\left(\beta\ \delta^H_{t-1}(x_t) + (1-\beta)\ \delta^{k^*}_t(x_t)\right)$$

where $\beta \in [0, 1]$ is set to 0.95 in our experiments. We have observed in our experiments that with this momentum-based attack strategy, model training is significantly stabilized and thus always converges. Meanwhile, the defender does not follow such a momentum-based strategy.[17] Moreover, we have also observed that replacing SGD with more advanced momentum-based optimization techniques (e.g., Adam [46]) also improves the stability of training. The experiments are thus conducted with Adam optimizers.

A.1.2. Network architectures and training settings

For MNIST, the target model is a *convolutional neural network* (CNN) with two convolutional layers (of 16 and 64 channels) and a fully-connected layer (of 64 hidden units). For CIFAR-10, the target model is a CNN with three convolutional layers (of 128, 128, and 512 channels) along with an average pooling layer (3×3) and a fully-connected layer (of 256 hidden units). For ImageNet, the target model is a CNN with four convolutional layers (of 64, 64, 64, and 512 channels and corresponding strides of 4, 3, 2, 1) along with an average pooling layer (7×7) and a fully-connected layer (of 128 hidden units).

For defense-GAN, defense-VAE, and defense-CVAE, the decoders/generative models are CNNs of 2 layers (of 32 and 16 channels). The encoders are CNNs of 2 layers (of 16 and 32 channels). The dimensions of the latent variables in these models are set to be 20. We have implemented a defence-CVAE baseline because this improved baseline will not only use the information of clean inputs, but also use the information of the labels of the inputs, which is also given in the known public validation set $\mathcal{D}_{\text{val}}$. The implementation of DC uses an auto-encoder architecture which is the same as the auto-decoder of the defense-VAE.

In Sec. 6.2, the subsampled MNIST dataset of digits 1 vs. 7 contains 40 training examples and 400 validation examples. The binary logistic regression classifier contains only a single linear layer. The CNN has two convolutional layers (of 16 and 16 channels) and a fully-connected layer (with 32 hidden units). We use such a small CNN because the calculation of Hessian is time-consuming if the target ML model has a large number of parameters.

The learning rate is 0.0008 for the Adam optimizer. The batch size is 500 for MNIST, 400 for CIFAR-10, and 100 for 2-class ImageNet. The test performances are recorded after 120 epochs for MNIST, 240 epochs for CIFAR-10, and 40 epochs for ImageNet. The non-linearity used is ReLU.

[17] The defender cannot follow such a momentum-based strategy because the attacks are adaptive in t. So, the defender cannot identify the same data inputs.

A.2. More experimental results

A.2.1. More analysis of synthetic experiment

From Theorem 1, the optimal R2T2 level-1 attack and defense strategies can be computed tractably as follows:

$$\delta_t^{1^*}(x_t) = \vec{\mathbf{e}}\left(-8\eta(\theta^* - \theta_t)^2\|z\|^2 x_t\right) = \vec{\mathbf{e}}\left(-x_t\right) ,$$
$$\sigma_t^{1^*}(x'_t) = \vec{\mathbf{e}}\left(8\eta(\theta^* - \theta_t)^2\|z\|^2 x'_t\right) = \vec{\mathbf{e}}\left(x'_t\right) .$$

From Theorem 2, the optimal R2T2 level-2 attack and defense strategies can be computed tractably as ϵ approaches 0:

$$\lim_{\epsilon\to 0}\delta_t^{2^*}(x_t) = \vec{\mathbf{e}}\left(-8\eta(\theta^* - \theta_t)^2\|z\|^2(1-2\eta\|x_t\|^2)x_t\right) = \vec{\mathbf{e}}\left(-(1-2\eta\|x_t\|^2)x_t\right) ,$$
$$\lim_{\epsilon\to 0}\sigma_t^{2^*}(x'_t) = \vec{\mathbf{e}}\left(8\eta(\theta^* - \theta_t)^2\|z\|^2(1-2\eta\|x'_t\|^2)x'_t\right) = \vec{\mathbf{e}}\left((1-2\eta\|x'_t\|^2)x'_t\right) .$$

Note that a perturbation along x_t will increase the model update step while a perturbation along $-x_t$ will decrease the model update step since $\|\theta_{t+1} - \theta_t\| \propto \|x''_t\|^2$. The difference from the level-1 players is that a level-2 player is aware of the case that a very large learning rate will cause the training to wiggle around the optimal parameter value. To see this, observe that an additional term appears in the level-2 strategies: $(1 - 2\eta\|x'_t\|^2)$. If $1 < 2\eta\|x'_t\|^2$, then this term will reverse the direction of the perturbation. This condition corresponds exactly to the situation where the SGD step will "overshoot" the optimal parameters θ^* (Fig. 3b). So, the level-2 strategy differs from the level-1 strategy in that the level-2 defense strategy will correctly reduce such overshooting while the level-2 attack strategy will correctly encourage it.

Nash equilibrium Since

$$\mathcal{U}_t(x''_t) = \left|\left[\theta_t + 2\eta(\theta^* - \theta_t)\|x''_t\|^2\right]z - \theta^* z\right|^2$$

is a polynomial of $\|x''_t\|$,

$$\lim_{\epsilon\to 0}\delta_t^{2^*}(x_t) = \vec{\mathbf{e}}\left(-8\eta(\theta^* - \theta_t)^2\|z\|^2(1-2\eta\|x_t\|^2)x_t\right) ,$$
$$\lim_{\epsilon\to 0}\sigma_t^{2^*}(x'_t) = \vec{\mathbf{e}}\left(8\eta(\theta^* - \theta_t)^2\|z\|^2(1-2\eta\|x'_t\|^2)x'_t\right) .$$

Also, as ϵ approaches 0, $x'_t = x_t$. Then, from the above, we know that $\lim_{\epsilon\to 0}\sigma_t^{2^*}(x'_t) = \vec{\mathbf{e}}((1 - 2\eta\|x'_t\|^2)x'_t) = \vec{\mathbf{e}}((1 - 2\eta\|x_t\|^2)x_t)$ is independent of the attack strategy; the special case of $x_t = 0$ is omitted from discussion since it leads to the same outcome. Therefore, as ϵ approaches 0, the optimal strategies of both players are independent of their opponents' strategies. Thus, the above optimal strategies form a Nash equilibrium. Note that the Nash equilibrium strategies are equal to the optimal level-2 strategies, which indicates that Nash equilibrium is attained in the case of a level-2 attacker against a level-2 defender; nevertheless, only level-$k \geq 3$ players are able to recognize such a Nash equilibrium when they find that their level-$(k-1)$ opponents are also playing Nash equilibrium strategies. Such a Nash equilibrium implies that all level-k ($k \geq 2$) strategies equal to the level-2 strategies.

With regards to the experimental settings, we set $\eta = 0.2$ and $\epsilon = 0.1$, and sample minibatch input x_t (of minibatch size 1) uniformly from the interval $(0, 2]$. We set $\theta^* = 20$ and $\theta_0 = -10$. We consider a single fixed point $z = 50$ as (the input of) the validation set.

A.2.2. Visualization of training-time adversarial examples

Fig. A.8 shows the visualization. The way we visualize a perturbation (possibly with negative value in it) is to normalize it to $[0, 1]$ by the scale of difference between its minimum and maximum pixel values.

On MNIST, it can be observed that the defender tends to perform defenses by removing attacker's perturbations (e.g., first two rows of Fig. A.8a) or mending the attacker's removals (e.g., digit '3' in the last row of Fig. A.8a). On CIFAR-10, the attacks and defenses, although still effective, are harder to interpret as they are abstract and result in visually indistinguishable adversarial examples/perturbed inputs x'_t and transformed inputs x''_t (Fig. A.8b). On 2-class ImageNet, although the attacks and defenses result in indistinguishable adversarial examples/perturbed inputs x'_t and transformed inputs x''_t, it can be observed that the defender tends to perform defenses by offsetting the adversarial attacks as the color contrasts between the attacker's perturbations and the defender's transformations in the first three rows of Fig. A.8c (respectively, purple vs. green, skyblue vs. red, blue vs. orange) correspond to negations of pixel values.

A.2.3. Non-gradient-based defense and obfuscated gradient defense

We additionally evaluate our R2T2 attack strategy against a non-gradient-based defense strategy like image transformation. Note from [47] that under a white-box test-time setting, even the most powerful image transformation (i.e., Crop

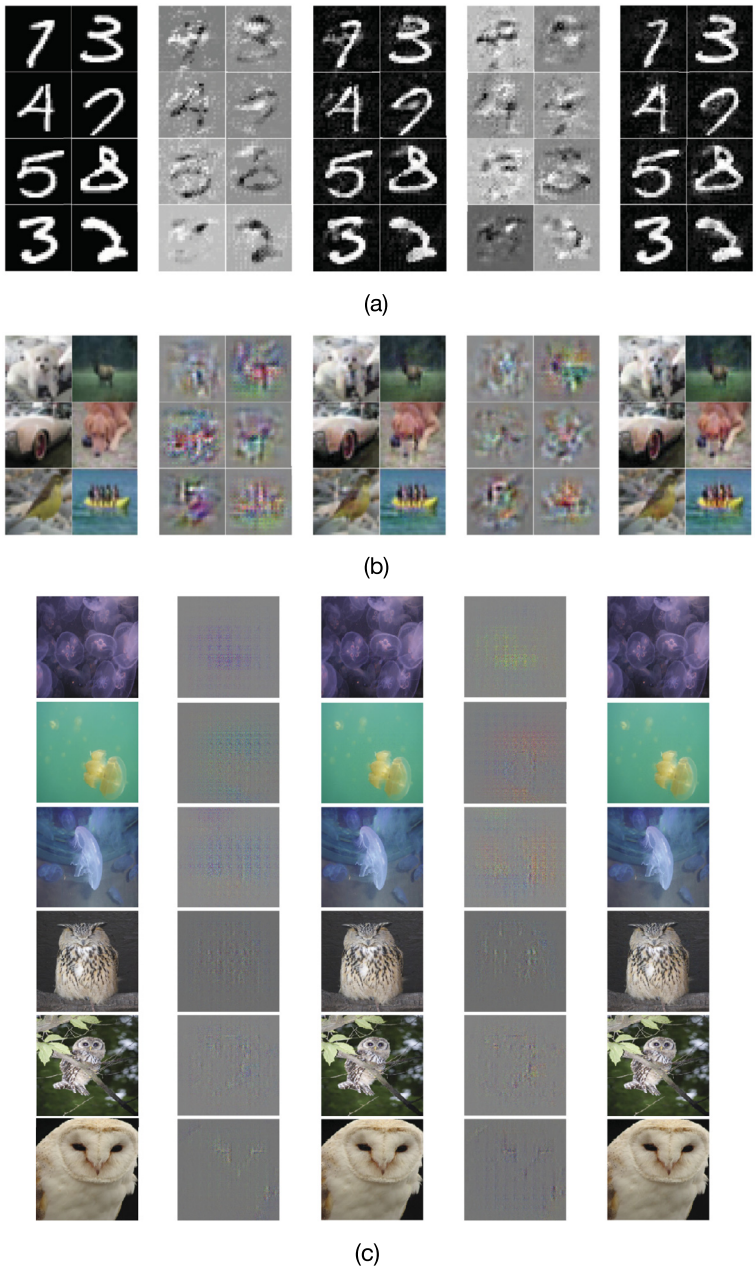


Fig. A.8. From left to right are, respectively, visualizations of the original sampled images x_t, perturbations $\delta_t^{1^*}(x_t)$ from level-1 attacks, adversarial examples/perturbed inputs x_t', transformations $\sigma_t^{1^*}(x_t')$ from level-1 defenses, and finally the transformed inputs x_t'' for model training on (a) MNIST, (b) CIFAR-10, and (c) ImageNet (color rescaled for viewing).

Ensemble) may not perform well against gradient-based attacks. Results in Table A.6 show that based on the same experimental setting in Sec 5.1, our level-1 attack strategy performs well against the non-gradient-based Crop Ensemble defense (i.e., same cropping procedure as that in [47]).

Our training-time attacks are essentially based on model gradients. It is known that gradient-based attack may be prone to obfuscated gradients [48]. In fact, the above-mentioned Crop Ensemble defense falls into the category of stochastic obfuscated gradients by randomizing the inputs before feeding them to the classifier. We additionally tested a case of obfuscated gradients by applying the *dropout* layer (with a dropout rate of 0.2) during model training s.t. the attacker would not know exactly which node will be dropped during training and inference. The results in Table A.6 show that our attack is still robust against obfuscated gradients. A possible explanation is that our validation set size is relatively large, so the attack perturbation of a particular training image is robust to a certain degree of obfuscated gradient. The degradation in attack

Table A.6
Mean test accuracy (%) (i.e., over 5 runs) for non-gradient-based Crop Ensemble defense [47] and obfuscated gradient-based Dropout defense against our level-1 attackers.

	MNIST	CIFAR-10
Crop Ensemble	13.0	22.1
Dropout	10.8	25.1

Table A.7
Mean test accuracy (%) $\pm$ 1 standard deviation (i.e., over 5 runs) for attacks (without a known validation set) against level-0 and level-1 defenders (with a known validation set).

Attacker \ Defender	MNIST		CIFAR-10	
	Level-0	Level-1	Level-0	Level-1
Level-1	14.0±4.6	96.0±1.2	18.9±1.7	50.1±7.2

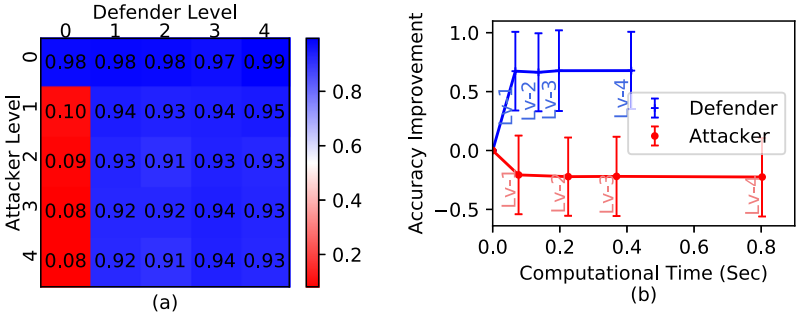


Fig. A.9. (a) Mean test accuracy (%) on MNIST. (b) Graph of mean accuracy improvement (over null strategy) vs. time needed to compute the strategies given a minibatch of MNIST.

performance in CIFAR-10 is more obvious than MNIST, likely due to more dropout layers (i.e., 5 compared to 3 in MNIST) in the deeper CNN model.

A.2.4. Evaluation of transferability: no known validation set for the attacker

We additionally examine a scenario when there is no known public validation set for the attacker. Note that the attacker can observe the clean minibatches and the attack strategy can be computed in a similar way by setting $\mathcal{D}_{\text{val}} = \mathcal{D}_t$ for the attacker.

The results in Table A.7 reveal a slight degradation in performance, possibly due to less training examples in a minibatch than in the validation set, hence yielding suboptimal attacks with large variances. However, the performance is not significantly different from that in the case of known validation set (Table 2), which means that the attacker may not need a known validation set to perform effective attacks.

A.2.5. Level-k strategies on MNIST

Fig. A.9a shows that when both players reason at level $k \geq 1$, the test accuracy is quite stable and does not change much with the players' reasoning levels. When both players reason beyond level $k \geq 1$, the change in the resulting test accuracy w.r.t. the players' reasoning level is small. The marginal benefit of reasoning at a higher level vs. time needed to compute the higher-level strategy in Fig. A.9b is similar to that in Fig. 7b on CIFAR-10.

A.2.6. Level-k strategies on 2-class ImageNet

Fig. A.10a (i.e., same as Fig. 7c) shows that level-$k \geq 1$ defenses are quite effective. Interestingly, as analyzed in Sec. 6.3, even when defending against a level-0 attacker (i.e., no attack), higher-level defenses still improves test accuracy noticeably.

A.2.7. Test-time robustness

Though our R2T2 framework has a clear distinction from test-time frameworks, we will also investigate the effect of our training procedure on the robustness w.r.t. test-time attacks. In particular, we consider the training setting of a level-1 defender against a weak attacker (i.e., either a level-0 attacker or a level-1 attacker with small ϵ_{atk}). Otherwise, the resulting model itself may already perform poorly on clean test inputs, not to mention perturbed test inputs. For test-time attacks, we have implemented YOPO-5-3 and FGSM-PGD with 5 PGD steps. The results are presented in Table A.8. Only limited improvements can be observed when a training-time attacker is introduced. After all, our framework is purely derived based on the training-time setting.

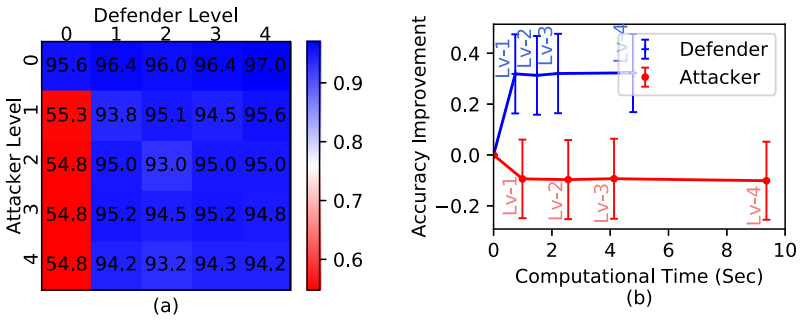


Fig. A.10. (a) Mean test accuracy on 2-class ImageNet (i.e., same as Fig. 7c in Sec. 6.3). (b) Graph of mean accuracy improvement (over null strategy) vs. time needed to compute strategies given a minibatch of 2-class ImageNet.

Table A.8
Mean test accuracy (%) on perturbed test inputs. We assume a *level*-1 *defender* during training. The second and sixth rows indicate *the type of attacker during training*. The first column indicates the type of attack used to generate the *test-time attack* perturbations together with the associated perturbation constraint.

MNIST		
	Level-0 attacker	Level-1 attacker ($\epsilon_{\text{atk}} = 0.5$)
YOPO (L_∞, $\epsilon = 0.3$)	1.3	3.0
FGSM-PGD (L_∞, $\epsilon = 0.3$)	1.6	5.6
CIFAR-10		
	Level-0 attacker	Level-1 attacker ($\epsilon_{\text{atk}} = 0.5$)
YOPO (L_∞, $\epsilon = 8/255$)	1.8	3.4
FGSM-PGD (L_∞, $\epsilon = 8/255$)	2.6	5.2

Appendix B. Proofs

B.1. Proof of Theorem 1

Proof. At level-1, the attacker best-responds to level-0 defender with only first-order gradient information (Assumption 1). While assuming higher-order gradients $\nabla_\theta^k \sum_{z\in\mathcal{D}_{\text{val}}^X} \mathcal{L}_{\theta_t}(z)$ are all zero for $k \geq 1$, the attacker's evaluation of the goal following a Taylor-series expansion is $\mathcal{U}_t = (\theta_{t+1} - \theta_t)^\top \nabla_\theta \sum_{z\in\mathcal{D}_{\text{val}}^X} \mathcal{L}_{\theta_t}(z)$.

Note that for level-1 attacker, $\mathcal{L}_{\theta_t}(x)$ is a linear function of x (Assumption 1) and thus $\nabla_\theta \mathcal{L}_{\theta_t}(x)$ is also a linear function of x. The attacker then adopts strategy

$$\begin{aligned}
\delta_t^{1^*}(x_t) &= \underset{\delta_t(x_t):\|\delta(x_t)\|\leq 1}{\arg\max} \; -[\eta \nabla_\theta \mathcal{L}_{\theta_t}(x_t + \epsilon\, \delta_t(x_t) + \epsilon\, \sigma_t^{0^*}(x_t'))]^\top \sum_{z\in\mathcal{D}_{\text{val}}^X} [\nabla_\theta \mathcal{L}_{\theta_t}(z)] \\
&= \underset{\delta_t(x_t):\|\delta_t(x_t)\|\leq 1}{\arg\max} \; -[\eta \nabla_\theta \mathcal{L}_{\theta_t}(x_t + \epsilon\, \delta_t(x_t))]^\top [\sum_{z\in\mathcal{D}_{\text{val}}^X} \nabla_\theta \mathcal{L}_{\theta_t}(z)] \\
&= -\vec{\mathbf{e}}\left(\eta \nabla_x [\nabla_\theta \mathcal{L}_{\theta_t}(x_t)]^\top [\sum_{z\in\mathcal{D}_{\text{val}}^X} \nabla_\theta \mathcal{L}_{\theta_t}(z)]\right).
\end{aligned} \tag{B.1}$$

where $\vec{\mathbf{e}}(\cdot)$ means taking the unit vector. A similar reasoning on the defender gives the optimal defense strategy on the theorem. □

Note that a level-0 strategy does not affect the functional form of level-1 strategy even if it is not null (e.g., $\sigma_t^{0^*}(x_t') \neq 0$): Assumption 1 has constrained that $\nabla_\theta \mathcal{L}_{\theta_t}(x_t)$ is a linear function of x_t. So, for the attacker,

$$\nabla_x \nabla_\theta \mathcal{L}_{\theta_t}(x_t + \epsilon\, \sigma_t^{0^*}(x_t')) = \nabla_x \nabla_\theta \mathcal{L}_{\theta_t}(x_t)$$

and vice versa for the defender. Consequently, level-0 strategy does not affect the functional form of level-k strategy for all $k \geq 1$.

B.2. Proof of Theorem 2

Proof. We first consider the attack strategy. Given Assumption 1, the second-order derivatives are available. We simplify the notation by denoting the optimal level-1 defense strategy $\sigma_t^{1^*}$ as σ, and denoting the optimal level-2 attack strategy $\delta_t^{2^*}$ as δ. $\delta(x_t)$ will result in the change of the stochastic gradient descent direction which can be expressed as a second-order polynomial of the strategy $\delta(x_t)$:

$$\begin{aligned}\delta\theta &= -\eta\nabla_\theta[\mathcal{L}_{\theta_t}(x_t+\epsilon\,\delta(x_t)+\epsilon\,\sigma(x_t'))-\mathcal{L}_{\theta_t}(x_t+\epsilon\,\sigma(x_t))]\\ &= -\eta\nabla_\theta[\epsilon\,\delta(x_t)^\top\nabla_x\mathcal{L}_{\theta_t}(x_t)]+\mathcal{O}(\epsilon^2)\,\mathbf{u}_1\end{aligned} \tag{B.2}$$

where $\mathbf{u}_1$ denotes a unit vector indicating the direction of residual term that we are not interested in. Let the domain of $\delta\theta$ be denoted by

$$\Theta=\{-\eta\nabla_\theta[\epsilon\,\delta(x_t)^\top\nabla_x\mathcal{L}_{\theta_t}(x_t)]+\mathcal{O}(\epsilon^2)\mathbf{u}_1\ :\ \|\delta(x_t)\|_2\le 1\}\,,$$

that is, $\delta\theta\in\Theta$. As $\epsilon\to 0$, $\lim_{\epsilon\to 0}\Theta=\Theta^0$ which means that $\delta\theta$ is bounded and varies within a line segment Θ^0 s.t. the domain Θ^0 is defined as

$$\Theta^0\triangleq\{c\,\eta\epsilon\nabla_\theta\|\nabla_x\mathcal{L}_{\theta_t}(x_t)\|_2\ :\ c\in[-1,1]\}\,.$$

Let $\Delta\theta\triangleq-\eta\nabla_\theta\mathcal{L}_{\theta_t}(x_t+\epsilon\,\sigma(x_t))$. The optimal gradient descent direction for the attacker can then be obtained by solving a constrained quadratic optimization problem since $\mathcal{L}_{\theta_{t+!}}(z)$ can be expanded as a quadratic polynomial of θ_{t+1} (Assumption 1):

$$\begin{aligned}\max_{\delta(x_t):\|\delta(x_t)\|_2\le 1}\mathcal{U}_t &= \max_{\delta\theta\in\Theta}\mathcal{U}_t\\ &= \max_{\delta\theta\in\Theta}\sum_{z\in\mathcal{D}_{\text{val}}^X}[\mathcal{L}_{\theta_t+\Delta\theta+\delta\theta}(z)-\mathcal{L}_{\theta_t}(z)]\\ &= \max_{\delta\theta\in\Theta}\sum_{z\in\mathcal{D}_{\text{val}}^X}\left[\delta\theta^\top\nabla_\theta\mathcal{L}_{\theta_t}(z)+\frac{1}{2}(\delta\theta+\Delta\theta)^\top H_{\theta_t|z}(\delta\theta+\Delta\theta)\right]\end{aligned} \tag{B.3}$$

where $H_{\theta_t|z}\triangleq\nabla_\theta^2\mathcal{L}_{\theta_t}(z)$, and $H_{\theta_t}\triangleq\sum_{z\in\mathcal{D}_{\text{val}}^X}H_{\theta_t|z}$. Given that Θ^0 is a linear constraint, we can transform the constrained quadratic optimization problem on $\delta\theta$ (B.3) into a constrained linear optimization problem on $\delta\theta$ as Θ approaches Θ^0 (ϵ approaches zero) since

$$\begin{aligned}&\lim_{\Theta\to\Theta^0}\arg\max_{\delta\theta\in\Theta}\sum_{z\in\mathcal{D}_{\text{val}}^X}[\delta\theta^\top\nabla_\theta\mathcal{L}_{\theta_t}(z)+\frac{1}{2}(\delta\theta+\Delta\theta)^\top H_{\theta_t|z}(\delta\theta+\Delta\theta)]\\ =&\lim_{\Theta\to\Theta^0}\arg\max_{\delta\theta\in\Theta}\ \delta\theta^\top\left\{\left[(\sum_{z\in\mathcal{D}_{\text{val}}^X}H_{\theta_t|z})\Delta\theta+\sum_{z\in\mathcal{D}_{\text{val}}^X}[\nabla_\theta\mathcal{L}_{\theta_t}(z)]\right]+\mathcal{O}(\|\delta\theta\|^2)\right\}\\ =&\lim_{\Theta\to\Theta^0}\arg\max_{\delta\theta\in\Theta}\ \delta\theta^\top\left[(\sum_{z\in\mathcal{D}_{\text{val}}^X}H_{\theta_t|z})\Delta\theta+\sum_{z\in\mathcal{D}_{\text{val}}^X}[\nabla_\theta\mathcal{L}_{\theta_t}(z)]\right]\\ =&\lim_{\Theta\to\Theta^0}\arg\max_{\delta\theta\in\Theta}\ \delta\theta^\top\mathcal{G}(x_t+\epsilon\,\sigma(x_t),\theta_t)\end{aligned} \tag{B.4}$$

where we define the function

$$\mathcal{G}(x,\theta_t)\triangleq-\eta(\sum_{z\in\mathcal{D}_{\text{val}}^X}H_{\theta_t|z})\nabla_\theta\mathcal{L}_{\theta_t}(x)+\sum_{z\in\mathcal{D}_{\text{val}}^X}[\nabla_\theta\mathcal{L}_{\theta_t}(z)] \tag{B.5}$$

for simplicity of notations. Since $\delta\theta$ is a second-order polynomial on $\delta(x_t)$ (B.2), the optimization problem (B.4) can now be turned into a quadratic constrained quadratic programming on $\delta(x_t)$. Now, the attacker can solve for its optimal strategy following (B.2) and (B.4):

$$
\begin{aligned}
&\lim_{\epsilon\to 0}\delta_t^{2^*}(x_t)\\
&=\lim_{\epsilon\to 0}\ \operatorname*{arg\,max}_{\delta(x_t):\|\delta(x_t)\|\le 1}\ \Big(-\eta\epsilon\ \delta(x_t)^\top[\nabla_x\nabla_\theta\mathcal{L}_{\theta_t}(x_t)]^\top+\mathcal{O}(\epsilon^2)\ \mathbf{u}_1^\top\Big)\ \mathcal{G}(x_t+\epsilon\ \sigma(x_t),\theta_t)\\
&=\lim_{\epsilon\to 0}\ \operatorname*{arg\,min}_{\delta(x_t):\|\delta(x_t)\|\le 1}\ \delta(x_t)^\top[\nabla_x\nabla_\theta\mathcal{L}_{\theta_t}(x_t)]^\top\mathcal{G}(x_t+\epsilon\ \sigma(x_t),\theta_t)\\
&=\operatorname*{arg\,min}_{\delta(x_t):\|\delta(x_t)\|\le 1}\ \delta(x_t)^\top[\nabla_x\nabla_\theta\mathcal{L}_{\theta_t}(x_t)]^\top\mathcal{G}(x_t,\theta_t)\\
&=-\vec{\mathbf{e}}([\nabla_x\nabla_\theta\mathcal{L}_{\theta_t}(x_t)]^\top\mathcal{G}(x_t,\theta_t))\ .
\end{aligned}
\tag{B.6}
$$

A similar reasoning on the defender side gives the optimal defense strategy on the theorem. □

B.2.1. Proof of Corollary 1

Proof. Following the above proof step in (B.3), $\sum_{z\in\mathcal{D}_{\text{val}}^X}\mathcal{L}_{\theta_t+\Delta\theta+\delta\theta}(z)$ can be expanded as a second-order polynomial of θ and is convex under the assumption of a positive definite H_{θ_t}.

Note that the OC attack solves

$$
\begin{aligned}
&\lim_{\epsilon\to 0}\max_{\delta(x_t):\|\delta(x_t)\|_2\le 1}\|\theta_{t+1}-\theta^\star\|=\lim_{\epsilon\to 0}\max_{\delta\theta\in\Theta}\|\theta_{t+1}-\theta^\star\|\\
&=\lim_{\Theta\to\Theta^0}\max_{\delta\theta\in\Theta}\|\theta_{t+1}-\theta^\star\|=\max_{\delta\theta\in\Theta^0}\|\theta_{t+1}-\theta^\star\|\ .
\end{aligned}
$$

In the above equation, $\delta\theta$ is defined in (B.2), and Θ and Θ^0 are defined above in the proof of Theorem 2 (Appendix B.2). Now,

$$
\begin{aligned}
&\max_{\delta\theta\in\Theta^0}\|\theta_{t+1}-\theta^\star\|\\
&=\max_{\delta\theta\in\Theta^0}\|\theta_t+\Delta\theta+\delta\theta-\theta^\star\|\\
&=\max_{\delta\theta\in\Theta^0}\Big(\delta\theta^\top[\theta_t+\Delta\theta-\theta^\star]+\mathcal{O}(\epsilon^2)\Big)\\
&=\max_{\delta\theta\in\Theta^0}\delta\theta^\top[\theta_t+\Delta\theta-\theta^\star]\\
&=\max_{\delta\theta\in\Theta^0}\delta\theta^\top\left[H_{\theta_t}^{-1}\nabla_\theta[\sum_{z\in\mathcal{D}_{\text{val}}^X}\mathcal{L}_{\theta_t+\Delta\theta}(z)]\right]\\
&=\max_{\delta\theta\in\Theta^0}\delta\theta^\top\left[H_{\theta_t}^{-1}\mathcal{G}(x_t+\epsilon\ \sigma(x_t),\theta_t)\right]
\end{aligned}
\tag{B.7}
$$

where $\Delta\theta=-\eta\nabla_\theta\mathcal{L}_{\theta_t}(x_t+\epsilon\ \sigma(x_t))$, which is defined earlier in Appendix B.2, is the direction of gradient descent for unattacked training inputs, the second last equality follows from the property of the second-order polynomial $\mathcal{L}$, and the last equality follows from the definition of $\mathcal{G}$ in (B.5). So, the final solution is

$$
\begin{aligned}
&\lim_{\epsilon\to 0}\ \operatorname*{arg\,max}_{\delta(x_t):\|\delta(x_t)\|_2\le 1}\ \|\theta_{t+1}-\theta^\star\|\\
&=\lim_{\epsilon\to 0}\ \operatorname*{arg\,max}_{\delta(x_t):\|\delta(x_t)\|_2\le 1}\ \Big(-\eta\epsilon\ \delta(x_t)^\top[\nabla_x\nabla_\theta\mathcal{L}_{\theta_t}(x_t)]^\top+\mathcal{O}(\epsilon^2)\ \mathbf{u}_1^\top\Big)\ H_{\theta_t}^{-1}\mathcal{G}(x_t+\epsilon\ \sigma(x_t),\theta_t)\\
&=-\vec{\mathbf{e}}([\nabla_x\nabla_\theta\mathcal{L}_{\theta_t}(x_t)]^\top H_{\theta_t}^{-1}\mathcal{G}(x_t,\theta_t))\ .
\end{aligned}
\tag{B.8}
$$

The defender's optimal strategy can be derived vice versa. □

B.3. Proof of Theorem 3

Note that for the problem of interest, input x varies within a closed domain bounded by the original input space and the perturbation constraint. Without loss of generality, we assume each dimension of x varies within $[0,1]$ (which is satisfied if we are dealing with normalized image input, otherwise, we normalize the bounded input space to $[0,1]$ and the analysis below will only differ by a normalizing constant).

Suppose the dimension of x is N. We denote the h-th order derivatives as $\mathcal{H}_{x_t}^h\triangleq\nabla_x^h\mathcal{U}_t(x_t)$, which is a h-th order N dimensional real *supersymmetric* tensor [49] for $h\ge 2$.

Lemma 1. *There exists a constant M s.t. under any level-k reasoning, the largest Z-eigenvalue*[18] *of $\mathcal{H}_x^h$ ($h \le k$) is not more than $M^h k!/(k-h)!$ for x in a closed domain $[0,1]^N$.*

Proof. First, note that the Z-eigenvalue always exists [49]. Under any finite level of reasoning, the Taylor expansion of $\mathcal{U}_t(x)$ on $x_0 = \mathbf{0}$ can be expressed as finite polynomial on $x_{\{1\}}, \ldots, x_{\{N\}}$ where the bracket denotes the dimensions of x.

$$\mathcal{U}_t(x) = \sum_{v_1,\ldots,v_N} C_{v_1,\ldots,v_N} x_{\{1\}}^{v_1} \ldots x_{\{N\}}^{v_N}.$$

We then denote $C_1 \triangleq \sum_{v_1,\ldots,v_N} |C_{v_1,\ldots,v_N}|$. Under level-$k$ reasoning, the loss function is a k-th order polynomial of x (Assumption 1). As a result, the Taylor expansion of $\mathcal{U}_t(x)$ on $x = 0$ is cut off in order k (a k-th order polynomial), and the above summation over $v_1, \ldots, v_N$ is constrained by $v_1 + \ldots + v_N \le k$. Note that the maximum absolute value of $x_{\{1\}}, \ldots, x_{\{N\}}$ is < 1 in such a closed domain, thus the scale of every element in the supersymmetric tensor $\mathcal{H}_x^h$ is bounded by $C_1 k!/(k-h)!$. The supersymmetric tensor $\mathcal{H}_x^h$ contains N^h elements, the Frobenius norm of $\mathcal{H}_x^h$ is thus bounded: $\|\mathcal{H}_x^h\|_F \le C_1 N^{h/2} k!/(k-h)!$. Since the largest absolute value of Z-eigenvalues of $\mathcal{H}_x^h$ is bounded by the Frobenius norm $\|\mathcal{H}_x^h\|_F$ (Theorem 9 in [50]), we can choose $M = C_1 N^{1/2}$ and arrive at the result. □

Lemma 2. *Under level-k reasoning, function $\mathcal{U}_t(x)$ is L-smooth w.r.t. x in the closed domain bounded by the perturbation constraint $X_t'' = \{x : \|x - x_t'\|_2 \le \epsilon\}$ where $L = M^2 k^2 (1 + 2M\epsilon)^{k-2}$ for some M.*

Proof. Note that the following optimization problem

$$\max_{\Delta, \|\Delta\|_2 \le 1} A \odot \Delta^h, \text{ where } A \odot x^h \triangleq \sum_{i_1,\ldots,i_h=1}^{N} a_{i_1,\ldots,i_h} x_{\{i_1\}} \cdots x_{\{i_h\}}, \tag{B.9}$$

where A is a h-th order N dimensional real supersymmetric tensor, has solution equals to the largest Z-eigenvalue of A [49]. Since $\forall \{x_1, x_1 + \Delta\} \in X_t''$ we have (note that this implies $\|\Delta\| < \varepsilon \triangleq 2\epsilon$):

$$\begin{aligned}
&\max_{\{x_1, x_1+\Delta\} \in X_t''} \frac{2}{\|\Delta\|^2} \left[\mathcal{U}_t(x_1 + \Delta) - \mathcal{U}_t(x_1) - (\nabla_x \mathcal{U}_t(x_1))^\top \Delta\right] \\
&\le \frac{2}{\|\Delta\|^2} \max_{\{x_1, x_1+\Delta\} \in X_t''} \sum_{h=2}^{k} \frac{1}{h!} \mathcal{H}_{x_1}^h \odot \Delta^h \\
&\le \sum_{h=2}^{k} \frac{2\,\varepsilon^{h-2} k!}{(k-h)! h!} M^h \\
&\le M^2 k^2 (1 + 2M\epsilon)^{k-2}
\end{aligned} \tag{B.10}$$

where the first inequality follows from Taylor expansion under level-k reasoning, the second inequality follows from Lemma 1, the third inequality holds since $\varepsilon \le 2\epsilon$ in the closed domain. The above inequality indicates that $\mathcal{U}_t(x)$ is L-smooth in the closed domain bounded by the perturbation constraint (X_t'') where $L = M^2 k^2 (1 + 2M\epsilon)^{k-2}$, according to the definition of L-smoothness. □

Lemma 3. *For a convex function $\mathcal{U}$ that is L-smooth, the iterates given by the projected gradient descent with step size $\Gamma = 1/L$ starting from x_0 satisfy for every step i*

$$|\mathcal{U}(x_{[i]}) - \mathcal{U}(x^*)| \le (2L/i) \|x_0 - x^*\|^2. \tag{B.11}$$

If $\mathcal{U}$ is further μ-strongly convex, we have

$$\|x_{[i]} - x^*\|^2 \le (1 - \mu/L)^i \|x_0 - x^*\|^2. \tag{B.12}$$

Proof. The lemma is a direct result of [41]. The proof is given in [41]. □

Proof of Theorem 3. Given Lemmas 2 and 3, we know that

$$|\mathcal{U}_t(x' + \epsilon\, \sigma_{[i]}) - \mathcal{U}_t(x_t' + \epsilon\, \sigma^{k^*}(x'))| \le (2L\epsilon^2/i) \|\sigma_{[0]} - \sigma_t^{k^*}(x_t')\|^2 \le 2\epsilon^2/(\Gamma_k i)\,. \tag{B.13}$$

[18] Refer to [49] for detailed definition of Z-eigenvalue.

If $\mathcal{U}_t(x)$ is μ-strongly convex, then

$$\|\sigma_{[i]} - \sigma_t^{k^*}(x_t')\|^2 \leq \frac{1}{\epsilon^2}(1-\mu/L)^j\epsilon^2\|\sigma_{[0]} - \sigma_t^{k^*}(x_t')\|^2 \leq (1-\Gamma_k\mu)^j \leq \exp(-\mu\Gamma_k j). \quad \square \tag{B.14}$$

B.3.1. Attacker PGD: the naive version

For the attacker, we define an auxiliary function $\mathcal{U}_t^\star(x) \triangleq \mathcal{U}_t\big(x + \epsilon\ \sigma_t^{(k-1)^*}(x)\big)$.

Corollary 2. ***(Convergence of naive attacker PGD)*** *Suppose that $-\mathcal{U}_t^\star(x)$ is a convex function of x and the 1-st to k-th gradients of $\sigma_t^{(k-1)^*}(x)$ w.r.t. x are finite. Then, there exists a constant M s.t. by setting the step size $\Gamma_k \triangleq 1/[M^2k^2(1+2M\epsilon)^{k-2}]$, to approximate level-$k$ strategy for input x_t, the PGD of $\delta_{[i]}$ starting from PGD step $i=0$ with $\delta_{[0]}=0$ is*

$$\overline{\delta}_{[i+1]} = \mathrm{Proj}\big(\delta_{[i]} + (\Gamma_k/\epsilon)\nabla_x\mathcal{U}_t^\star(x'_{[i]})\big) \tag{B.15}$$

where $x'_{[i]} \triangleq x_t + \epsilon\ \delta_{[i]}$, has following convergence guarantee in any PGD step i:

$$\big|\mathcal{U}_t^\star\big(x_t + \epsilon\ \delta_{[i]}\big) - \mathcal{U}_t^\star\big(x_t + \epsilon\ \delta_t^{k^*}(x_t)\big)\big| \leq 2\epsilon^2/(\Gamma_k i)\ . \tag{B.16}$$

If $-\mathcal{U}_t^\star(x)$ is further μ-strongly convex,

$$\big\|\delta_{[i]} - \delta_t^{k^*}(x_t)\big\| \leq \exp(-\mu\Gamma_k i)\ . \tag{B.17}$$

Proof. Its proof is similar to that of Theorem 3. $\square$

Convergence of attacker's NPGD: The naive attacker PGD (B.15) cannot be computed in practice because we cannot compute $\sigma_t^{(k-1)^*}$ tractably. However, if $\sigma_{[i]}$ in the attacker NPGD (9) approximates $\sigma_t^{(k-1)^*}(x'_{[i]})$ perfectly without error in every PGD step i, the NPGD is then identical to the naive attacker PGD, thus $\delta_{[i]}$ of the NPGD enjoy the convergence guarantee on Corollary 2 above.

B.3.2. Estimation of M and Γ_k

To estimate M from training, note that from the above proof of Theorem 3 we can see that $N^{1/2}|\mathcal{U}_t([1,\ldots,1])| = N^{1/2}|\sum_{v_1,\ldots,v_N} C_{v_1,\ldots,v_N}| \leq C_1 N^{1/2} = M$ provides a lower bound of M. Although in theory, this lower bound can be loose, we have found that estimating M with this lower bound works well in practice s.t. we can always observe improvements when reasoning into a higher level. To calculate this lower bound, we train the ML model on a white image ($[1,\ldots,1]$) for 1 iteration and calculate the change of validation loss. As a result, the estimation of M is roughly 0.203, 0.351, and 1.04 for MNIST, CIFAR-10 and 2-class ImageNet, respectively.

The step size Γ_k can thus be calculated correspondingly. Note that in practice it is always the case that $(\Gamma_k/\epsilon)|\nabla_x\mathcal{U}_t(x)| > 1$, thus we can safely set the step size to positive infinity and the resulting attacker NPGD/defender PGD still performs well. To estimate a reasonable number of total PGD steps, we first ensure that approximating level-k strategies at least needs to perform k steps of PGD. We then set the total number i of steps as $2\Gamma_k$, $0.5\Gamma_k$, and $0.01\Gamma_k$ for, respectively, MNIST, CIFAR-10, and 2-class ImageNet with the estimation of M above.

B.4. Generalization to more than one example in $\mathcal{D}_t$

We extend the notation of $\mathcal{U}_t$ s.t. $\mathcal{D}_t'' \triangleq \{[x''_{jt}, y(x_{jt})]\}_{j=1,\ldots}$ is the minibatch after the perturbations of the attacker and the defender, $\mathcal{D}_t''^X \triangleq \{[x''_{jt}]\}_{j=1,\ldots}$ is the collection of the inputs of $\mathcal{D}_t''$, $\mathcal{U}_t(\mathcal{D}_t''^X)$ represents the utility function if current training in conducted on the (perturbed) minibatch $\mathcal{D}_t''$, and $\mathcal{U}_t(x_t'')$ still represents the utility function if current training in conducted on the (perturbed) single data input x_t''. As a result, the base game can be written as

$$\begin{aligned} &\textbf{Attacker:} \max_{\delta_t(x_t):\|\delta_t(x_t)\|\leq 1} \mathcal{U}_t(\mathcal{D}_t''^X)\ , \\ &\textbf{Defender:} \max_{\sigma_t(x_t'):\|\sigma_t(x_t')\|\leq 1} -\mathcal{U}_t(\mathcal{D}_t''^X)\ . \end{aligned} \tag{B.18}$$

Suppose that now, the minibatch data $\mathcal{D}_t$ can contain more than one training example s.t. $\mathcal{D}_t^X = \{[x_{jt}] : j = 1, 2, \ldots\}$. Then,

$$\begin{aligned} \frac{\partial\mathcal{U}_t(\mathcal{D}_t''^X)}{\partial x_j} =& -\eta[\frac{\partial}{\partial x_j}\nabla_\theta \sum_{x''_{lt}\in\mathcal{D}_t''^X}\mathcal{L}_{\theta_t}(x''_{lt})]^\top \sum_{z\in\mathcal{D}_{\mathrm{val}}^X}[\nabla_\theta\mathcal{L}_{\theta_t}(z)] \\ =& -\eta[\frac{\partial}{\partial x_j}\nabla_\theta\mathcal{L}_{\theta_t}(x''_{jt})]^\top \sum_{z\in\mathcal{D}_{\mathrm{val}}^X}[\nabla_\theta\mathcal{L}_{\theta_t}(z)] \\ =& \nabla_x\mathcal{U}_t(x''_{jt}) \end{aligned} \tag{B.19}$$

where the last equality follows from (7).

B.4.1. Level-1 strategies

Due to (B.19), by following the proof in Appendix B.1, we can show that the level-1 strategies still follow (6) and remain unchanged.

B.4.2. Level-2 strategies

Due to (B.19), it is not hard to follow the proof in Appendix B.2 and show that the level-2 strategies in the limit of small ϵ should be changed to that below:

Corollary 3. ***(Level-2 strategies)*** *Given a training minibatch $\mathcal{D}_t^X = \{[x_{jt}] : j = 0, 1, \ldots\}$, the optimal level-2 attack and defense strategies in the limit of small ϵ are*

$$\begin{aligned}
\lim_{\epsilon\to 0} \delta_t^{2^*}(x_t) &= -\vec{\mathbf{e}}\big([\nabla_x \nabla_\theta \mathcal{L}_{\theta_t}(x_t)]^\top \big[\sum_{z\in\mathcal{D}_{val}^X} \nabla_\theta \mathcal{L}_{\theta_t}(z) - \eta H_{\theta_t} \nabla_\theta \sum_{x_{jt}\in\mathcal{D}_t^X} \mathcal{L}_{\theta_t}(x_{jt}) \big]\big), \\
\lim_{\epsilon\to 0} \sigma_t^{2^*}(x'_t) &= \vec{\mathbf{e}}\big([\nabla_x \nabla_\theta \mathcal{L}_{\theta_t}(x'_t)]^\top \big[\sum_{z\in\mathcal{D}_{val}^X} \nabla_\theta \mathcal{L}_{\theta_t}(z) - \eta H_{\theta_t} \nabla_\theta \sum_{x'_{jt}\in\mathcal{D'}_t^X} \mathcal{L}_{\theta_t}(x'_{jt}) \big]\big).
\end{aligned} \tag{B.20}$$

B.4.3. Level-k strategies

Due to (B.19), both defender PGD (10) and attacker NPGD (9) remain unchanged.

References

[1] N. Jones, Computer science: the learning machines, Nature 505 (2014) 146–148.
[2] H.B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, S. Chikkerur, D. Liu, M. Wattenberg, A.M. Hrafnkelsson, T. Boulos, J. Kubica, Ad click prediction: a view from the trenches, in: Proc. KDD, 2013, pp. 1222–1230.
[3] A. Pal, R. Vidal, A game theoretic analysis of additive adversarial attacks and defenses, in: Proc. NeurIPS, 2020.
[4] I. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in: Proc. ICLR, 2015.
[5] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, in: Proc. ICLR, 2014.
[6] J. Feng, Q.Z. Cai, Z.H. Zhou, Learning to confuse: generating training time adversarial data with auto-encoder, in: Proc. NeurIPS, 2019, pp. 11971–11981.
[7] T.-H. Ho, C. Camerer, K. Weigelt, Iterated dominance and iterated best response in experimental "p-beauty contests", Am. Econ. Rev. 88 (4) (1998) 947–969.
[8] R. Nagel, Unraveling in guessing games: an experimental study, Am. Econ. Rev. 85 (5) (1995) 1313–1326.
[9] D.O. Stahl, P.W. Wilson, Experimental evidence on players' models of other players, J. Econ. Behav. Organ. 25 (3) (1994) 309–327.
[10] D.O. Stahl, P.W. Wilson, On players' models of other players: theory and experimental evidence, Games Econ. Behav. 10 (1) (1995) 218–254.
[11] C.F. Camerer, T.-H. Ho, J.-K. Chong, A cognitive hierarchy model of games, Q. J. Econ. 119 (3) (2004) 861–898.
[12] D. Gill, V. Prowse, Cognitive ability, character skills, and learning to play equilibrium: a level-k analysis, J. Polit. Econ. 124 (6) (2016) 1619–1676.
[13] Y. Jin, Does level-k behavior imply level-k thinking?, Available at SSRN, https://ssrn.com/abstract=3138321, 2018.
[14] N. Carlini, D. Wagner, Towards evaluating the robustness of neural networks, in: Proc. IEEE S&P, 2017, pp. 39–57.
[15] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, in: Proc. ICLR, 2018.
[16] D. Zhang, T. Zhang, Y. Lu, Z. Zhu, B. Dong, You only propagate once: accelerating adversarial training via maximal principle, in: Proc. NeurIPS, 2019, pp. 227–238.
[17] X. Li, S. Ji, Defense-VAE: a fast and accurate defense against adversarial attacks, in: Proc. ECML/PKDD, 2019, pp. 191–207.
[18] D. Meng, H. Chen, MagNet: a two-pronged defense against adversarial examples, in: Proc. CCS, 2017, pp. 135–147.
[19] P. Samangouei, M. Kabkab, R. Chellappa, Defense-GAN: protecting classifiers against adversarial attacks using generative models, in: Proc. ICLR, 2018.
[20] A. Kurakin, I. Goodfellow, S. Bengio, Adversarial machine learning at scale, in: Proc. ICLR, 2017.
[21] S. Singla, S. Feizi, Second-order provable defenses against adversarial attacks, in: Proc. ICML, 2020.
[22] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, P. McDaniel, Ensemble adversarial training: attacks and defenses, in: Proc. ICLR, 2018.
[23] M. Kearns, M. Li, Learning in the presence of malicious errors, SIAM J. Comput. 22 (4) (1993) 807–837.
[24] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E.C. Lupu, F. Roli, Towards poisoning of deep learning algorithms with back-gradient optimization, in: Proc. AISec, 2017, pp. 27–38.
[25] P.W. Koh, P. Liang, Understanding black-box predictions via influence functions, in: Proc. ICML, 2017, pp. 1885–1894.
[26] M. Barreno, B. Nelson, A.D. Joseph, J.D. Tygar, The security of machine learning, Mach. Learn. 81 (2) (2010) 121–148.
[27] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, B. Li, Manipulating machine learning: poisoning attacks and countermeasures for regression learning, in: Proc. IEEE S&P, 2018, pp. 19–35.
[28] J. Steinhardt, P.W. Koh, P.S. Liang, Certified defenses for data poisoning attacks, in: Proc. NeurIPS, 2017, pp. 3517–3529.
[29] M. Brückner, T. Scheffer, Stackelberg games for adversarial prediction problems, in: Proc. SIGKDD, 2011, pp. 547–555.
[30] B. Li, Y. Vorobeychik, Feature cross-substitution in adversarial classification, in: Proc. NeurIPS, 2014.
[31] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, V. Shmatikov, How to backdoor federated learning, in: Proc. AISTATS, 2020, pp. 2938–2948.
[32] T. Gu, K. Liu, B. Dolan-Gavitt, S. Garg, BadNets: evaluating backdooring attacks on deep neural networks, IEEE Access 7 (2019) 47230–47244.
[33] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, B.Y. Zhao, Neural cleanse: identifying and mitigating backdoor attacks in neural networks, in: Proc. IEEE S&P, 2019, pp. 707–723.
[34] Z. Dai, Y. Chen, B.K.H. Low, P. Jaillet, T.-H. Ho, R2-B2: recursive reasoning-based Bayesian optimization for no-regret learning in games, in: Proc. ICML, 2020, pp. 2291–2301.
[35] T. Pang, X. Yang, Y. Dong, H. Su, J. Zhu, Accumulative poisoning attacks on real-time data, in: Proc. NeurIPS, vol. 34, 2021, pp. 2899–2912.
[36] X. Zhang, X. Zhu, L. Lessard, Online data poisoning attacks, in: Proc. L4DC, PMLR, 2020, pp. 201–210.
[37] N. Papernot, P. McDaniel, I. Goodfellow, Transferability in machine learning: from phenomena to black-box attacks using adversarial samples, arXiv: 1605.07277, 2016.

[38] B. Li, C. Chen, W. Wang, L. Carin, Second-order adversarial attack and certifiable robustness, arXiv:1809.03113, 2018.
[39] T. Miyato, S. Maeda, M. Koyama, S. Ishii, Virtual adversarial training: a regularization method for supervised and semi-supervised learning, IEEE Trans. Pattern Anal. Mach. Intell. 41 (8) (2019) 1979–1993.
[40] S. Baluja, I. Fischer, Learning to attack: adversarial transformation networks, in: Proc. AAAI, 2018, pp. 2687–2695.
[41] Y. Nesterov, Lectures on Convex Optimization, Springer, 2018.
[42] S. Chaudhury, H. Roy, S. Mishra, T. Yamasaki, Adversarial training time attack against discriminative and generative convolutional models, IEEE Access 9 (2021) 109241–109259.
[43] S.A. Rebuffi, S. Gowal, D.A. Calian, F. Stimberg, O. Wiles, T. Mann, Fixing data augmentation to improve adversarial robustness, arXiv:2103.01946, 2021.
[44] J. Ho, A. Jain, P. Abbeel, Denoising diffusion probabilistic models, in: Proc. NeurIPS, 2020, pp. 6840–6851.
[45] S. Yun, D. Han, S.J. Oh, S. Chun, J. Choe, Y. Yoo, Cutmix: regularization strategy to train strong classifiers with localizable features, in: Proc. ICCV, 2019, pp. 6023–6032.
[46] D.P. Kingma, J. Ba Adam, A method for stochastic optimization, arXiv:1412.6980, 2014.
[47] C. Guo, M. Rana, M. Cisse, L. van der Maaten, Countering adversarial images using input transformations, in: Proc. ICLR, 2018.
[48] A. Athalye, N. Carlini, D. Wagner, Obfuscated gradients give a false sense of security: circumventing defenses to adversarial examples, in: Proc. ICML, 2018, pp. 274–283.
[49] L. Qi, H. Chen, Y. Chen, Tensor Eigenvalues and Their Applications, Springer, 2018.
[50] L. Qi, Rank and eigenvalues of a supersymmetric tensor, the multivariate homogeneous polynomial and the algebraic hypersurface it defines, J. Symb. Comput. 41 (12) (2006) 1309–1327.