# Perturbation Analysis of Learning Algorithms: Generation of Adversarial Examples from Classification to Regression

Emilio Rafael Balda, Arash Behboodi, *Member, IEEE,* and Rudolf Mathar, *Member, IEEE*

***Abstract*—Despite the tremendous success of deep neural networks in various learning problems, it has been observed that adding intentionally designed adversarial perturbations to inputs of these architectures leads to erroneous classification with high confidence in the prediction. In this work, we show that adversarial examples can be generated using a generic approach that relies on the perturbation analysis of learning algorithms. Formulated as a convex program, the proposed approach retrieves many current adversarial attacks as special cases. It is used to propose novel attacks against learning algorithms for classification and regression tasks under various new constraints with closed-form solutions in many instances. In particular, we derive new attacks against classification algorithms which are shown to be top-performing on various architectures. Although classification tasks have been the main focus of adversarial attacks, we use the proposed approach to generate adversarial perturbations for various regression tasks. Designed for single pixel and single subset attacks, these attacks are applied to autoencoding, image colorization and real-time object detection tasks, showing that adversarial perturbations can degrade equally gravely the output of regression tasks[1].**

## I. Introduction

Deep Neural Networks (DNNs) excelled in recent years in many learning tasks and demonstrated outstanding achievements in speech analysis [1] and visual tasks [2]–[5]. Despite their success, they have been shown to suffer from instability in their classification under adversarial perturbations [6]. Adversarial perturbations are intentionally worst-case designed noises that aim at changing the output of a DNN to an incorrect one. The explosion of research during past years makes it almost impossible to refer to all important works in this area and do justice to all excellent works. However, we refer to several important results from the literature, that are highly connected to this paper.

Although DNNs might achieve robustness to random noise [7], it was shown that there is a clear distinction between the robustness of a classifier to random noise and its robustness to adversarial perturbations. The existence of adversarial perturbations was known for machine learning algorithms [8], however, they were first noticed in deep learning research in [6]. The peculiarity of adversarial perturbations lied in the fact that they managed to fool state-of-the-art networks into making confident and wrong decisions in classification tasks, and they, nevertheless, appeared unperceived to the naked eye. These discoveries gave rise to extensive research on understanding the instability of DNNs, exploring various attacks and devising multiple defenses (for instance refer to [9]–[11] and references therein). Most adversarial attacks fall generally into two classes, white-box and black-box attacks. White-box attacks use complete or partial knowledge of the machine learning architecture, see for instance [12]. In constrast, black-box attacks do not require any information about the target neural network, see for instance [13]. In this work, we focus only on white-box attacks with full knowledge of the function implemented by the learning algorithm. The attacks, as in [12], [14], [15], act on the system inputs and add perturbations that are not perceived by the system's administrator such that the performance of the system is severely degraded.

Adversarial perturbations were obtained in [6] to maximize the prediction error at the output and were approximated using box-constrained Limited memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm. The Fast Gradient Sign Method (FGSM) in [12] was based on finding the scaled sign of the gradient of the cost function. Note that the FGSM aims at minimizing $\ell_\infty$-norm of the perturbation while the former algorithm minimizes $\ell_2$-norm of the perturbation under box constraint on the perturbed example.

More effective attacks utilize either iterative procedures or randomization of the perturbations and instances as in [14], [16]. The algorithm DeepFool [14] conducts an iterative linearization of the DNN to generate perturbations that are minimal in the $\ell_p$-norm for $p > 1$. In [17] the authors propose an iterative version of FGSM, called Basic Iterative Method (BIM). This method was later extended in [16], where randomness was introduced in the computation of adversarial perturbations. This attack is called the Projected Gradient Descent (PGD) method and was employed in [16] to devise a defense against adversarial examples. An iterative algorithm based on PGD combined with randomization was introduced in [18] and has been used to dismantle many defenses so far [19]. Another popular way of generating adversarial examples is by constraining the $\ell_0$-norm of the perturbation. These types of attacks are known as single pixel attacks [20] and multiple pixel attacks [21]. Despite their differences, many of the above attacks share a similar underlying principle. At some moment, they rely on a simple characterization of input perturbations on the output of the algorithm. In this paper, we build on this idea to design

Institute for Theoretical Information Technology (TI), RWTH Aachen University.

[1]In the spirit of encouraging reproducible research, the implementations used in this paper have been made available at:

github.com/ebalda/adversarialconvex

new attacks for classification and regression tasks.

An interesting feature of these perturbations is their generalization over other datasets and DNNs [12], [15]. These perturbations are called universal adversarial perturbations. This is partly explained by the fact that certain underlying properties of the perturbation, such as orientation in case of image perturbation, matters the most and is therefore generalized through different datasets. For example, the attack from [22] shows that adversarial examples transfer from one random instance of a neural network to another. In that work, the authors showed the effectiveness of these types of attacks for enhancing the robustness of neural networks, since they provide diverse perturbations during adversarial training. Moreover, [23] showed the existence of universal adversarial perturbations that are independent from the system and the target input. We will not go into more details about universal perturbations in this paper.

Since the rise of adversarial examples for image classification, novel algorithms have been developed for attacking other types of systems. In the field of computer vision, [24] constructed an attack on image segmentation, while [25] designed attacks for object detection. The Houdini attack [26] aims at distorting speech recognition systems. Moreover, [27] taylored an attack for recurrent neural networks, and [28] for reinforcement learning. Adversarial examples exist for probabilistic methods as well. For instance, [29] showed the existence of adversarial examples for generative models. For regression problems, [30] designed an attack that specifically targets variational autoencoders. It seems, however, to be a gap in literature when it comes to adversarial attacks on regression problems. One of the goals of this paper is to fill this gap by proposing attacks on various regression problems.

Before going further, we shortly overview some theoretical explanations of adversarial examples as well as common defenses. There are various theories regarding the nature of adversarial examples and the subject is heavily investigated. Initially, the authors in [12] proposed the linearity hypothesis where the existence of adversarial images is attributed to the approximate linearity of classifiers, although this hypothesis has been challenged in [31]. Some other theories focus mostly on decision boundaries of classifiers and their analytic properties [7], [32]. The work from [33] provides a framework for determining the robustness of a classifier against adversarial examples with some performance guarantees. For a more recent theoretical approach to this problem refer to [34]. There exist several types of defenses against adversarial examples, as well as subsequent methods for bypassing them. Defensive distillation was proposed in [35] as a defense method. It extracts knowledge about the training points and feeds it back as part of the training to improve the robustness of the network. The method directly controls the amplitude of network gradients as a defense technique. The authors in [36], however, proposed three attacks to bypass defensive distillation. Similarly, the attacks from [18], bypassed 7 out of 9 non-certified defenses presented at ICLR 2018 that claimed to be white-box secure. The term non-certified refers to defenses that do not provide any certificate that guarantees robustness against particular attacks. The most common defense is adding adversarial examples to the training set, also known as adversarial training. For that purpose different adversarial attacks may be employed. Recently, training with the PGD attack is used in [16] to provide the state-of-the-art defense against adversarial examples for various image classification datasets.

### A. Our Contribution

In this work, we provide various new adversarial attacks for classification and regression tasks. We consider only white-box attacks where the full knowledge of the machine learning model is assumed to be available. The common thread of these new attacks is the perturbation analysis of learning algorithms that yields a tractable optimization problem for generating new attacks. This idea, as it will be shown, underlies many existing attacks. We build upon our previous work [37] to introduce a connection between perturbation analysis of learning algorithms and adversarial examples. For classification tasks, we start with a fairly general formulation of the adversarial generation problem, namely (AGP), and show how perturbation analysis can be used to obtain a convex optimization problem for generating adversarial examples. This problem, (AGP.II), is the basis for generating adversarial examples. We address feasibility issues of the preceding problem and propose multiple techniques to get around this issue in general, some of which are introduced for the first time. Besides, closed form solutions are provided for a few cases. We propose novel adversarial attacks for classification, given in Algorithm 1, which are benchmarked with state-of-the-art attacks. Our proposed attack is an iterative method that constraints the norm of adversarial perturbations and apply it after randomization of the training instance.

Another contribution of this paper is to use a similar technique in context of adversarial perturbations for regression problems, a topic that has not been yet widely explored. Regression loss functions differ from classification loss functions in that it is sufficient to maximize the output perturbation measured by an application-dependent function, for instance the $\ell_2$-norm of the output error. In classification tasks, such maximization might not necessarily change the output label particularly because these perturbations might push the instances far away from classification margins. There is, however, no natural margin in regression tasks. We use perturbation analysis to formulate the loss maximization problem in a tractable fashion. Similar to classification problems, adversarial examples can be generated using convex optimization with closed-form solutions for a few special cases. The proposed optimization problems for regression are, to the best of our knowledge, novel. We discuss single pixel and single subset attacks for regression tasks. It is shown that this problem is related to the MaxCut problem and hence difficult to solve. We propose a greedy algorithm to solve this problem.

Finally, the proposed algorithms are experimentally evaluated using state-of-the-art benchmarks for classification and regression tasks. In classification tasks, the performance of our proposed attack is consistently among the top attacks and sometimes the best one. In regression tasks, we demonstrate several attacks for regression tasks such as image colorization,

autoencoding and object detection. Although autoencoders show better robustness in general, the other algorithms are severely degraded under adversarial perturbations.

### B. Notation

Throughout this work, the letters $a, b, \ldots$ are used for scalars, $\mathbf{a}, \mathbf{b}, \ldots$ for vectors, $\mathbf{A}, \mathbf{B}, \ldots$ for matrices and $\mathcal{A}, \mathcal{B}, \ldots$ for sets. To denote the set $\{1, \ldots, n\}$, we make use of the shorthand notation $[n]$ for $n \in \mathbb{N}$. The operator $\|\cdot\|_p$ denotes $\ell_p$-norm of a vector with $p \geq 1$, while $(\cdot)^\top$ the matrix transposition.

## II. Fooling Classifiers with First-Order Perturbation Analysis

The perturbation analysis, also called sensitivity analysis, is used in signal processing for analytically quantifying the error at the output of a system that occurs as consequence of a known perturbation at the system's input. Adversarial images can also be considered as a slightly perturbed version of original images that manage to change the output of the classifier. Indeed, the generation of adversarial examples in [12], [14] is implicitly based on maximizing the effect of an input perturbation on a relevant function which is either the classifier function or the cost function used for training. In the FGSM, given in [12], the perturbation at the output of the training cost function is first analyzed using first-order perturbation analysis of the cost function and then maximized to fool the algorithm. The DeepFool method, given in [14], maximizes the output perturbation for the linearized approximation of the underlying classifier which is indeed its first order-perturbation analysis. In this section, we develop further the connection between perturbation analysis and adversarial examples. To generate adversarial examples, we provide a generic approach that starts from a fairly abstract formulation and transforms it into a tractable problem by sequentially using perturbation analysis.

### A. Adversarial Perturbation Design Problem

We start with formulating the problem of adversarial perturbation design. As it was mentioned above, adversarial examples can be considered as a perturbed version of training examples changed by an adversarial perturbation $\boldsymbol{\eta}$. As we will see later, the perturbation analysis is straightforward when the underlying system behaves smoothly and can be modeled by differentiable functions. The classifier function, however, maps inputs to discrete set of labels and therefore it is not differentiable. Instead, the classification problem is slightly modified as follows to facilitate the perturbation analysis.

**Definition 1** (Classification)**.** *A classifier is defined by the mapping $k : \mathbb{R}^M \to [K]$ that maps an input $\mathbf{x} \in \mathbb{R}^M$ to its estimated class $k(\mathbf{x}) \in [K]$. The mapping $k(\cdot)$ is itself defined by*

$$k(\mathbf{x}) = \underset{l \in [K]}{\operatorname{argmax}} \{f_l(\mathbf{x})\}, \tag{1}$$

*where $f_l(\mathbf{x}) : \mathbb{R}^M \to \mathbb{R}$'s are called score functions representing the probability of class belonging.*

The function $f(\mathbf{x})$ given by the vector $(f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))$ can be assumed to be differentiable almost everywhere for many classifiers.

The problem of adversarial generation consists of finding a perturbation that changes the classifier's output. However, it is desirable for adversarial perturbations to modify training instances only in an insignificant and unnoticeable way. This is controlled by adding a constraint on the adversarial perturbation. For instance, the perturbation generated by the FGSM is bounded in the $\ell_\infty$-norm and the DeepFool method directly minimizes the norm of the perturbation that changes the classifier's output. While DeepFool might generate perturbations that are perceptible, the FGSM might not change the classifier's output. It is indeed an intriguing property of adversarial examples that the perturbation does not distort the image significantly so that the naked eye can not detect any notable change in the images. One way of imposing this property in adversarial design is to constrain the input perturbation to keep the output of the ground truth classifier, also called oracle classifier [10], intact. The oracle classifier represents the naked eye in case of image classification. The score functions of the oracle classifier are denoted by $g_l(\cdot)$. The undetectability constraint for an adversarial perturbation $\boldsymbol{\eta}$ is formulated as

$$L_g(\mathbf{x}, \boldsymbol{\eta}) = g_{k(\mathbf{x})}(\mathbf{x} + \boldsymbol{\eta}) - \max_{l \neq k(\mathbf{x})} g_l(\mathbf{x} + \boldsymbol{\eta}) > 0\,. \tag{2}$$

This inequality means that even after applying the perturbation $\boldsymbol{\eta}$ to an instance $\mathbf{x}$ with the correct label $k(\mathbf{x})$, the score function of the correct class $g_{k(\mathbf{x})}(\mathbf{x} + \boldsymbol{\eta})$ is superior to the other score functions. A fundamental trait of the oracle classifier, therefore, is its robustness to the adversarial perturbation $\boldsymbol{\eta}$. Therefore the problem of adversarial design can be formulated as follows.

**Problem 1** (Adversarial Generation Problem)**.** *For a given $\mathbf{x} \in \mathbb{R}^M$, find a perturbation $\boldsymbol{\eta} \in \mathbb{R}^M$ to fool the classifier $k(\cdot)$ by the adversarial sample $\widehat{\mathbf{x}} = \mathbf{x} + \boldsymbol{\eta}$ such that $k(\mathbf{x}) \neq k(\widehat{\mathbf{x}})$ and the oracle classifier is not changed, i.e.,*

$$\begin{aligned} \text{Find}: \quad & \boldsymbol{\eta} \\ \text{s.t.} \quad & L_f(\mathbf{x}, \boldsymbol{\eta}) = f_{k(\mathbf{x})}(\mathbf{x} + \boldsymbol{\eta}) - \max_{l \neq k(\mathbf{x})} f_l(\mathbf{x} + \boldsymbol{\eta}) < 0 \\ & L_g(\mathbf{x}, \boldsymbol{\eta}) = g_{k(\mathbf{x})}(\mathbf{x} + \boldsymbol{\eta}) - \max_{l \neq k(\mathbf{x})} g_l(\mathbf{x} + \boldsymbol{\eta}) > 0 \end{aligned} \tag{AGP}$$

We pose the problem (AGP) as a general starting point for the adversarial design. This problem consists on finding a perturbation that changes the classifier's output while the class of input remains the same to the oracle. As it is, the problem (AGP) does not have that much utility in practice but serves as the starting point. Next we explore different methods for making this problem tractable, all of them based on perturbation analysis of constraints. Since $\mathbf{x}$ and $f$ are fixed for the attacker, we simplify the notation by dropping the subscript $f$ and assuming that gradients are always with respect to $\boldsymbol{\eta}$, that is $L(\mathbf{x}, \cdot) = L_f(\mathbf{x}, \cdot)$ and $\nabla L(\mathbf{x}, \cdot) = \nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \cdot)$. We keep these shorthand notations throughout the paper.

### B. Perturbation Analysis

The problem (AGP) constitutes the starting point for adversarial design. Two issues prevent us from directly applying it. We

use perturbation analysis to propose a solution to these issues. First, the oracle function is not known in general. However, since the oracle function is assumed to be robust to adversarial perturbations, the constraint can be replaced with constraints on the perturbation itself, for instance by imposing upper bounds on the $\ell_p$-norm of the perturbation. Indeed, if the perturbation analysis is applied to the oracle function, its robustness implies that the output perturbation is $\mathcal{O}(\|\boldsymbol{\eta}\|_p)$. Therefore, adding constraints on the $\ell_p$-norm translates into constraints on the oracle function as in (2). The constraint on the oracle function can be therefore approximated by $\|\boldsymbol{\eta}\|_p \leq \varepsilon$ for sufficiently small $\varepsilon \in \mathbb{R}^+$. This means that the noise is sufficiently small in $\ell_p$-norm sense so that the observer does not notice it. Different classes of attacks can be obtained for different choices of $p$ and are well known in the literature such as $\ell_\infty$-attacks, $\ell_2$-attacks and $\ell_1$-attacks (see the survey in [9] for details).

The second problem with (AGP) is that the function $L(\mathbf{x}, \cdot)$ can be non-convex or difficult to optimize directly. Here again, perturbation analysis can be employed to approximate $L(\mathbf{x}, \cdot)$ with a tractable function like linear functions. The first order perturbation analysis of $L$ yields

$$L(\mathbf{x}, \boldsymbol{\eta}) = L(\mathbf{x}, \mathbf{0}) + \boldsymbol{\eta}^\top \nabla L(\mathbf{x}, \mathbf{0}) + \mathcal{O}(\|\boldsymbol{\eta}\|_2^2),$$

where $\mathcal{O}(\|\boldsymbol{\eta}\|_2^2)$ contains higher order terms. Following the above approximations using perturbation analysis, we propose the following relaxed optimization problem.

**Problem 2** (Relaxed Adversarial Generation Problem)**.** *For a given $\mathbf{x} \in \mathbb{R}^M$, a perturbation $\boldsymbol{\eta} \in \mathbb{R}^M$ is found to fool the classifier $k(\cdot)$ using*

$$\begin{aligned} &\textit{Find:} \quad \boldsymbol{\eta} \\ &\textit{s.t.} \quad L(\mathbf{x}, \mathbf{0}) + \boldsymbol{\eta}^\top \nabla L(\mathbf{x}, \mathbf{0}) < 0, \quad \|\boldsymbol{\eta}\|_p \leq \varepsilon. \end{aligned} \tag{AGP.II}$$

The above problem is the result of applying perturbation analysis to the problem (AGP). It is a convex optimization problem that can be efficiently solved. One significant advantage of the problem (AGP.II) is that it can incorporate additional constraints to the optimization problem such as sparsity constraint that leads to single-pixel attacks. Furthermore, as we will see later, this formulation of the problem yields some of the well known existing adversarial methods. Unfortunately, the above problem is not always feasible as stated in the following proposition.

**Theorem 1.** *The optimization problem* (AGP.II) *is not feasible if for $q = \frac{p}{p-1}$*

$$\varepsilon \|\nabla L(\mathbf{x}, \mathbf{0})\|_q < L(\mathbf{x}, \mathbf{0}). \tag{3}$$

*Proof.* The proof follows a simple duality argument and is an elementary optimization theory result. A similar result can be inferred from [38]. We repeat the proof for completeness. Note that the dual norm of $\ell_p$ is defined by

$$\|\mathbf{x}\|_p^* = \sup\{\mathbf{a}^\top \mathbf{x} : \|\mathbf{a}\|_p \leq 1\}.$$

Furthermore $\|\mathbf{x}\|_p^* = \|\mathbf{x}\|_q$ for $q = \frac{p}{p-1}$. Since the $\ell_p$-norm of $\boldsymbol{\eta}$ is bounded by $\varepsilon$, the value of $\boldsymbol{\eta}^\top \nabla L(\mathbf{x}, \mathbf{0})$ is always bigger than $-\varepsilon \|\nabla L(\mathbf{x}, \mathbf{0})\|_p^*$. However if the condition (3) holds, then we have

$$L(\mathbf{x}, \mathbf{0}) + \boldsymbol{\eta}^\top \nabla L(\mathbf{x}, \mathbf{0}) \geq L(\mathbf{x}, \mathbf{0}) - \varepsilon \|\nabla L(\mathbf{x}, \mathbf{0})\|_p^* > 0.$$

Therefore, the problem is not feasible. □

Theorem 1 shows that given a vector $\mathbf{x}$, the adversarial perturbation should have at least $\ell_p$-norm equal to $\frac{L(\mathbf{x},\mathbf{0})}{\|\nabla L(\mathbf{x},\mathbf{0})\|_q}$. In other words if the ratio $\frac{L(\mathbf{x},\mathbf{0})}{\|\nabla L(\mathbf{x},\mathbf{0})\|_q}$ is small, then it is easier to fool the network by the $\ell_p$-attacks. In that sense, Theorem 1 provides an insight into the stability of classifiers. In [14], the authors suggest that the robustness of the classifiers can be measured as

$$\widehat{\rho}_1(f) = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \frac{\|\widehat{\mathbf{r}}(\mathbf{x})\|_p}{\|\mathbf{x}\|_p},$$

where $\mathcal{D}$ denotes the test set and $\widehat{\mathbf{r}}(\mathbf{x})$ is the minimum perturbation required to change the classifier's output. The above theorem suggests that one can also use the following as the measure of robustness, which was also derived in [38]

$$\widehat{\rho}_2(f) = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \frac{L(\mathbf{x}, \mathbf{0})}{\|\nabla L(\mathbf{x}, \mathbf{0})\|_q}.$$

The lower $\widehat{\rho}_2(f)$, the easier it gets to fool the classifier and therefore it becomes less robust to adversarial examples. One can also look at other statistics related to $\frac{L(\mathbf{x},\mathbf{0})}{\|\nabla L(\mathbf{x},\mathbf{0})\|_q}$ in order to evaluate the robustness of classifiers.

To get around the feasibility problem, a first approach is to start with a small enough $\varepsilon$ and iteratively solve the problem (AGP.II). Other methods consist of relaxing one of the constraints while keeping the other constraint intact. We will explicate these methods in the next section.

### C. Feasible Adversarial Perturbation Designs

Theorem 1 shows that the optimization problem (AGP.II) might not be feasible. We propose to get around this issue by solving an optimization problem which keeps only one of the constraints, depending on the scenario, and selects an appropriate objective function to preserve the other constraint as much as possible. The objective function in this sense models the deviation from the constraint and is minimized in the optimization problem. We consider two optimization problems for this purpose.

First, the norm-constraint on the perturbation is preserved. The following optimization problem, called gradient-based norm-constrained method (GNM), aims at minimizing $L(\mathbf{x}, \mathbf{0}) + \boldsymbol{\eta}^\top \nabla L(\mathbf{x}, \mathbf{0})$ by solving the following problem:

$$\min_{\boldsymbol{\eta}} \left\{ L(\mathbf{x}, \mathbf{0}) + \boldsymbol{\eta}^\top \nabla L(\mathbf{x}, \mathbf{0}) \right\} \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_p \leq \varepsilon\,. \tag{4}$$

This method finds the best perturbation under the norm-constraint and is a novel formulation to the best of our knowledge. The constraint aims at guaranteeing that the adversarial images are still imperceptible by an ordinary observer. Note that (4) is fundamentally different from [14], [38], where the norm of the noise does not appear as a constraint. Using a similar duality argument, the problem (4) has a closed form solution given below.

**Theorem 2.** *If* $\nabla L(\mathbf{x},\boldsymbol{\eta}) = (\frac{\partial L(\mathbf{x},\boldsymbol{\eta})}{\partial \eta_1},\ldots,\frac{\partial L(\mathbf{x},\boldsymbol{\eta})}{\partial \eta_M})$*, the closed form solution to the minimizer of the problem* (4) *is given by*

$$\boldsymbol{\eta} = -\varepsilon \frac{1}{\|\nabla L(\mathbf{x},\mathbf{0})\|_q^{q-1}} \operatorname{sign}(\nabla L(\mathbf{x},\mathbf{0})) \odot |\nabla L(\mathbf{x},\mathbf{0})|^{q-1} \tag{5}$$

*for* $q = \frac{p}{p-1}$*, where* $\operatorname{sign}(\cdot)$ *and* $|\cdot|^{q-1}$ *are applied element-wise, and* $\odot$ *denotes the element-wise (Hadamard) product. Particularly for* $p = \infty$*, we have* $q = 1$ *and the solution is given by the following*

$$\boldsymbol{\eta} = -\varepsilon \operatorname{sign}(\nabla L(\mathbf{x},\mathbf{0}))\,. \tag{6}$$

*Proof.* Based on the duality argument from convex analysis, it is known that

$$\sup_{\|\boldsymbol{\eta}\|_p \le 1} \boldsymbol{\eta}^\top \nabla L(\mathbf{x},\mathbf{0}) = \|\nabla L(\mathbf{x},\mathbf{0})\|_p^*,$$

where $\|\cdot\|^*$ is the dual norm. This implies that the objective function is lower bounded by $L(\mathbf{x},\mathbf{0}) - \varepsilon\|\nabla L(\mathbf{x},\mathbf{0})\|_p^*$. It is easy to verify that the minimum is attained by (5). □

The advantage of these convex relaxations, apart from enjoying computationally efficient solutions, is that one can incorporate other convex constraints into the optimization problem to guarantee additional required properties of the perturbation. Note that the introduced method in (4) can also be used for other target functions or learning problems. If the training cost function is maximized under a norm constraint, as in [12], the solution of (4) with $p = \infty$ recovers the adversarial perturbations obtained via the FGSM. The problem (4) guarantees that the perturbation is small, however, it might not change the classifier's output.

The second optimization problem (AGP.II), on the other hand, preserves the constraint for changing the classifier's output and minimizes the perturbation norm instead. The feasibility problem of (AGP.II) can therefore be simplified to

$$\min_{\boldsymbol{\eta}} \|\boldsymbol{\eta}\|_p \quad \text{s.t.} \quad L(\mathbf{x},\mathbf{0}) + \boldsymbol{\eta}^\top \nabla L(\mathbf{x},\mathbf{0}) \le 0\,, \tag{7}$$

which recovers the result in [38] and in [14] although without the iterative procedure. This problem has a similar closed form solution.

**Proposition 1.** *If* $\nabla L(\mathbf{x},\boldsymbol{\eta}) = (\frac{\partial L(\mathbf{x},\boldsymbol{\eta})}{\partial \eta_1},\ldots,\frac{\partial L(\mathbf{x},\boldsymbol{\eta})}{\partial \eta_M})$*, the closed form solution to the problem* (7) *is given by*

$$\boldsymbol{\eta} = -\frac{L(\mathbf{x},\mathbf{0})}{\|\nabla L(\mathbf{x},\mathbf{0})\|_q^{q-1}} \operatorname{sign}(\nabla L(\mathbf{x},\mathbf{0})) \odot |\nabla L(\mathbf{x},\mathbf{0})|^{q-1} \tag{8}$$

*for* $q = \frac{p}{p-1}$.

Note that the perturbation found in Proposition 1, like the solution to GNM, aligns with the gradient of the classifier function and they only differ in their norm. Although the perturbation in (8), unlike the solution to GNM, is able to fool the classifier, the perturbation in (8) might be perceptible by the oracle classifier.

The formulations (4) and (7) do not exhaust all possibilities for solving the feasibility problem above. There are other variants of adversarial generation methods that rely on an implicit perturbation analysis of a relevant function. These methods can be easily obtained by small modification of the methods above.

Iterative procedures can be easily adapted to the current formulation by repeating the optimization problem until the classifier output changes while keeping the perturbation small enough at each step such that the problem (AGP.II) remains feasible. Later we provide an iterative version of the GNM and compare it with DeepFool [14], as well as other methods.

Another class of methods relies on introducing randomness in the generation process. We call this technique dithering. A notable example is the PGD attack introduced in [16] which is one of the state-of-the-art attacks. The first-order approximation is then taken around another point $\widetilde{\boldsymbol{\eta}}$ with $\widetilde{\varepsilon} \triangleq \|\widetilde{\boldsymbol{\eta}}\|_p \le \varepsilon$. In other words we approximate $L(\mathbf{x},\cdot)$ by a linear function around the point $\widetilde{\boldsymbol{\eta}}$ within an $\widetilde{\varepsilon}$-radius from $\boldsymbol{\eta} = \mathbf{0}$. This new point $\widetilde{\boldsymbol{\eta}}$ can be computed at random using arbitrary distributions with $\ell_p$-norm bounded by $\varepsilon$. Changing the center of the first order approximation from $\mathbf{0}$ to $\widetilde{\boldsymbol{\eta}}$ does not change the nature of the problem since $L(\mathbf{x},\boldsymbol{\eta}) \approx L(\mathbf{x},\widetilde{\boldsymbol{\eta}}) + (\boldsymbol{\eta} - \widetilde{\boldsymbol{\eta}})^\top \nabla L(\mathbf{x},\widetilde{\boldsymbol{\eta}})$ leads to the following problem

$$\min_{\boldsymbol{\eta}} L(\mathbf{x},\widetilde{\boldsymbol{\eta}}) + (\boldsymbol{\eta} - \widetilde{\boldsymbol{\eta}})^\top \nabla L(\mathbf{x},\widetilde{\boldsymbol{\eta}}) \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_p \le \varepsilon,$$

which is equivalent to:

$$\min_{\boldsymbol{\eta}} \boldsymbol{\eta}^\top \nabla L(\mathbf{x},\widetilde{\boldsymbol{\eta}}) \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_p \le \varepsilon\,. \tag{9}$$

From this result one can add randomness to the computation of adversarial examples by selecting $\widetilde{\boldsymbol{\eta}}$ in a random fashion. This is desirable when training models with adversarial examples since it increases the diversity of the adversarial perturbations during training ( See [22] where the authors introduce the Randomized Fast Gradient Sign Method (R-FGSM) attack).

The summary of our proposed approach is as follows. We start with the problem (AGP) by determining an appropriate loss function $L(\mathbf{x},\cdot)$. Next the problem is simplified to a tractable problem like (AGP.II) using perturbation analysis. Additional constraints on the perturbation are added at will. If the problem is feasible, the final solution yields the adversarial perturbation. Otherwise, one can use iterative and randomization techniques explained above or solve problems (4) or (7). We follow similar steps to generate adversarial examples for regression problems in the next section.

## III. From Classification to Regression

In classical statistical learning theory, regression problems are defined in the following manner. Given $N \in \mathbb{N}$ samples $\{(\mathbf{x}_i,\mathbf{y}_i)\}_{i=1}^N$ drawn according to some unknown distribution $P_{X,Y}$, a regression model computes a function $f : \mathbb{R}^M \to \mathbb{R}^K$ that aims to minimize the expected loss $\mathbb{E}_P(\mathcal{L}(f(\mathbf{x}),\mathbf{y}))$, where $\mathcal{L} : \mathbb{R}^M \times \mathbb{R}^K \to \mathbb{R}$ is a function that measures the difference between $f(\mathbf{x})$ and $\mathbf{y}$. While logarithmic losses are popular in classification problems, the squared loss $\mathcal{L}(f(\mathbf{x}),\mathbf{y}) = \|f(\mathbf{x}) - \mathbf{y}\|_2^2$ is mostly used for the general regression setting. There is, however, no general natural loss function for the regression problems, although each problem disposes of specific appropriate choices. Squared loss is certainly suitable for

function approximation tasks and in particular autoencoders. Peak Signal to Noise Ratio (PSNR) is suitable for measuring the quality of image outputs. Throughout this paper, it is assumed that the loss function is properly chosen for the adversarial attack on the underlying regression problem. For the sake of notation, given $\mathbf{y}$ and $f$, let us redefine $L(\mathbf{x}, \boldsymbol{\eta})$ as $L(\mathbf{x}, \boldsymbol{\eta}) = \mathcal{L}(f(\mathbf{x}+\boldsymbol{\eta}), \mathbf{y})$.

For a given $f$, $\mathbf{x}$ and $\mathbf{y}$, an adversarial attacker finds an additive perturbation vector $\boldsymbol{\eta}$ that is *imperceptible* to the administrator of the target system, while maximizing the loss of the perturbed input $L(\mathbf{x}, \boldsymbol{\eta})$ as

$$\max_{\boldsymbol{\eta}} L(\mathbf{x}, \boldsymbol{\eta}) \quad \text{s.t.} \quad \boldsymbol{\eta} \text{ is imperceptible}\,.$$

In contrast with classification problems where maximum perturbations at the output might not change the class, adversarial instances maximize the output perturbation in regression problems.

As in (4), a constraint on the $\ell_p$-norm of $\boldsymbol{\eta}$ models imperceptibility leading to the following formulation of the problem

$$\max_{\boldsymbol{\eta}} \|\mathbf{y} - f(\mathbf{x}+\boldsymbol{\eta})\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_p \leq \varepsilon\,. \tag{10}$$

Consider the image colorization problem where the goal is to add proper coloring on top of gray scale images. In this problem, $f(\cdot)$ is the regression algorithm and assumed to be known however the ground truth colorization $\mathbf{y}$ is generally unknown. Without knowing $\mathbf{y}$, the optimization problem (10) is ill posed and cannot be solved in general. There are some cases where the output $\mathbf{y}$ is known by the nature of the problem, for instance, when $f(\cdot)$ is an encoder-decoder pair as in autoencoders for which $\mathbf{y} = \mathbf{x}$.

Since the goal is to perturb the acting regression algorithm, we can assume that $\mathbf{y} \approx f(\mathbf{x})$ which means that the algorithm provides a good although not perfect approximation of the ground truth function. We use the formulation in (10) and discuss the implications of applying the approximation $\mathbf{y} \approx f(\mathbf{x})$ in later sections.

### A. A Quadratic Programming Problem

In general $f(\mathbf{x})$ is a non-linear and non-convex function, so we have that $L(\mathbf{x}, \cdot)$ is non-convex. Here again the perturbation analysis of $f(\cdot)$ can be used to relax (10) and to obtain a convex formulation of the adversarial problem. The first order perturbation analysis of $f(\mathbf{x})$ yields the approximation $f(\mathbf{x}+\boldsymbol{\eta}) \approx f(\mathbf{x}) + \mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}$, where $\mathbf{J}_f(\cdot)$ is the Jacobian matrix of $f(\cdot)$. This approximation leads to the following convex approximation of $L(\mathbf{x}, \cdot)$:

$$\begin{aligned} &L(\mathbf{x}, \boldsymbol{\eta}) \\ &\quad \approx \|\mathbf{y}\|_2^2 - 2\mathbf{y}^\top(f(\mathbf{x}) + \mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}) + \|f(\mathbf{x}) + \mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}\|_2^2 \\ &\quad = \|\mathbf{y}\|_2^2 - 2\mathbf{y}^\top f(\mathbf{x}) + \|f(\mathbf{x})\|_2^2 \\ &\qquad + 2\left(f(\mathbf{x}) - \mathbf{y}\right)^T \mathbf{J}_f(\mathbf{x})\boldsymbol{\eta} + \|\mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}\|_2^2\,. \end{aligned}$$

Since the first three terms of this expression do not depend on $\boldsymbol{\eta}$, the optimization problem from (10) is reduced to

$$\max_{\boldsymbol{\eta}} 2\left(f(\mathbf{x}) - \mathbf{y}\right)^T \mathbf{J}_f(\mathbf{x})\boldsymbol{\eta} + \|\mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_p \leq \varepsilon\,. \tag{11}$$

The above convex maximization problem is, in general, challenging and NP-hard. Nevertheless, since $\mathbf{y}$ is usually not known, we may use the assumption that $\mathbf{y} \approx f(\mathbf{x})$, which simplifies the problem to

$$\max_{\boldsymbol{\eta}} \|\mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_p \leq \varepsilon\,. \tag{12}$$

Although this problem is a convex quadratic maximization under an $\ell_p$-norm constraint and in general challenging, it can be solved efficiently in some cases. For general $p$, the maximum value is indeed related to the operator norm of $\mathbf{J}_f(\mathbf{x})$ [39]. This norm is central in stability analysis of many signal processing algorithms (for instance see [40]). The operator norm of a matrix $\mathbf{A} \in \mathbb{C}^{m\times n}$ between $\ell_p$ and $\ell_q$ is defined as

$$\|\mathbf{A}\|_{p\to q} \triangleq \sup_{\|\mathbf{x}\|_p \leq 1} \|\mathbf{A}\mathbf{x}\|_q.$$

Using this notion, we can see that $\|\frac{\boldsymbol{\eta}}{\varepsilon}\|_p \leq 1$ leads to $\|\mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}\|_2 = \varepsilon\|\mathbf{J}_f(\mathbf{x})\frac{\boldsymbol{\eta}}{\varepsilon}\|_2 \leq \varepsilon\|\mathbf{J}_f(\mathbf{x})\|_{p\to 2}$. Therefore, the problem of finding a solution to (12) amounts to finding the operator norm $\|\mathbf{J}_f(\mathbf{x})\|_{p\to 2}$. First observe that the maximum value is achieved on the border namely for $\|\boldsymbol{\eta}\|_p = \varepsilon$. In the case where $p = 2$, this problem has a closed-form solution. If $\mathbf{v}_{\max}$ is the unit $\ell_2$-norm eigenvector corresponding to the maximum eigenvalue of $\mathbf{J}_f(\mathbf{x})^\top \mathbf{J}_f(\mathbf{x})$, then

$$\boldsymbol{\eta}^* = \pm\varepsilon\, \mathbf{v}_{\max} \tag{13}$$

solves the optimization problem. The maximum eigenvalue of $\mathbf{J}_f(\mathbf{x})^\top \mathbf{J}_f(\mathbf{x})$ corresponds to the square of the spectral norm $\|\mathbf{J}_f(\mathbf{x})\|_{2\to 2}$.

Another interesting case is when $p = 1$. In general, the $\ell_1$-norm is usually used as a regularization technique to promote sparsity. When the solution of a problem should satisfy a sparsity constraint, the direct introduction of this constraint into the optimization leads to NP-hardness of the problem. Instead the constraint is relaxed by adding $\ell_1$-norm regularization. The adversarial perturbation designed in this way tends to have only a few non-zero entries. This corresponds to scenarios like single pixel attacks where only a few pixels are supposed to change. For this choice, we have

$$\|\mathbf{A}\|_{1\to 2} = \max_{k\in[n]} \|\mathbf{a}_k\|_2,$$

where $\mathbf{a}_k$'s are the columns of $\mathbf{A}$. Therefore, if the columns of the Jacobian matrix are given by $\mathbf{J}_f(\mathbf{x}) = [\mathbf{J}_1 \dots \mathbf{J}_M]$, then

$$\|\mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}\|_2 \leq \varepsilon \max_{k\in[M]} \|\mathbf{J}_k\|_2,$$

and the maximum is attained with

$$\boldsymbol{\eta}^* = \pm\varepsilon\mathbf{e}_{k^*} \quad \text{for} \quad k^* = \operatorname*{argmax}_{k\in[M]} \|\mathbf{J}_k\|_2, \tag{14}$$

where the vector $\mathbf{e}_i$ is the $i$-th canonical vector. For the case of gray-scale images, where each pixel is represented by a single entry of $\mathbf{x}$, this constitutes a single pixel attack. Some additional constraints must be added in the case of RGB images, where each pixel is represented by a set of three values.

Finally, the case where the adversarial perturbation is bounded with the $\ell_\infty$-norm is also of particular interest. This bound guarantees that the noise entries have bounded values.

The problem of designing adversarial noise corresponds to finding $\|\mathbf{J}_f(\mathbf{x})\|_{\infty\to 2}$. Unfortunately, this problem turns out to be NP-hard [41]. However, it is possible to approximate this norm using semi-definite programming as proposed in [42]. Semi-definite programming scales badly with input dimension in terms of computational complexity, namely $O(n^6)$ with $n$ the underlying dimension, and therefore might not be suitable for fast generation of adversarial examples when the input dimension is very high. We address these problems later in Section IV, where we obtain fast approximate solutions for $\|\mathbf{J}_f(\mathbf{x})\|_{\infty\to 2}$ and single pixel attacks.

### B. A Linear Programming Problem

The methods derived in Section III-A suffer from one main drawback, they require storing $\mathbf{J}_f(\mathbf{x}) \in \mathbb{R}^{K\times M}$ into memory. While this may be doable for some applications, it is not feasible for others. For example, if the target system is an autoencoder for RGB images with size $680 \times 480$, that is $M = K = 680\cdot 480\cdot 3 \approx 9\cdot 10^5$, storing $\mathbf{J}_f(\mathbf{x}) \in \mathbb{R}^{9\cdot 10^5\times 9\cdot 10^5}$ requires loading around $8\cdot 10^{11}$ values into memory, which is in most cases not tractable. Note that, in order to solve (12) for $p = 2$, we would require computing the eigenvalue decomposition of $\mathbf{J}_f(\mathbf{x})^\top \mathbf{J}_f(\mathbf{x})$ as well. This motivates us to relax the problem into a linear programming problem as in Section II, where $\mathbf{J}_f(\mathbf{x})$ is computed implicitly and we do not require to store it. To that end, we relax (10) by directly applying a first order approximation of $L$, that is $L(\mathbf{x},\boldsymbol{\eta}) \approx L(\mathbf{x},\mathbf{0}) + \boldsymbol{\eta}^\top \nabla L(\mathbf{x},\mathbf{0})$ . Using this approximation the problem from (10) is now simplified to

$$\max_{\boldsymbol{\eta}} \nabla L(\mathbf{x},\mathbf{0})^\top \boldsymbol{\eta} \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_p \le \varepsilon\,, \tag{15}$$

where $\nabla L(\mathbf{x},\mathbf{0}) = -2\mathbf{J}_f(\mathbf{x})^T(\mathbf{y} - f(\mathbf{x}))$. Note that the attacks discussed in Section II for classification follow the same formulation with another choice of $L(\mathbf{x},\cdot)$. Therefore, the closed-form solution of (15) can be obtained from (5).

Unfortunately using $\mathbf{y} \approx f(\mathbf{x})$ yields zero gradient in (15), thus leaving this approximation useless for obtaining adversarial perturbations. This problem is tackled by taking the approximation around another random point $\widetilde{\boldsymbol{\eta}}$ within and $\widetilde{\varepsilon}$-ball radius from $\boldsymbol{\eta} = \mathbf{0}$ as in (9), with $\widetilde{\varepsilon} \le \varepsilon$. As it was mentioned above, this dithering mechanism is also used in classification problems for instance in [16]. Since the problem obtained by applying this dithering technique on (15) is equivalent to (9), regardless of the function $L$, the results derived for classification (see Section II-C) can be reused to solve (15).

## IV. Single Subset Attacks

Another popular way of modeling undetectability, in the field of image recognition, is by constraining the number of pixels that can be modified by the attacker. This gave birth to single and multiple pixel attacks. Note that, for the case of gray-scale images, the solutions obtained in (14) and (5) provide already single pixels attacks. This is not true for RGB images where each pixel is represented by a subset of three values. Since our analysis is not limited to image based systems, we refer to these type of attacks which target only a subset of entries as single subset attacks.

Since perturbations belong to $\mathbb{R}^M$, let us partition $[M] = \{1,\dots,M\}$ into $S$ possible subsets $\mathcal{S}_1,\dots,\mathcal{S}_S$. The sets can in general have different cardinalities. However, we assume here that all of them have the same cardinality of $Z = M/S$, where $\mathcal{S}_s = \{i_s^1,\dots,i_s^Z\} \subseteq [M]$. We define the mixed zero-$\mathcal{S}$ norm $\|\cdot\|_{0,\mathcal{S}}$ of a vector, for the partition $\mathcal{S} = \{\mathcal{S}_1,\dots,\mathcal{S}_S\}$, as the number of subsets containing at least one index associated to a non-zero entry of $\mathbf{x}$[2]:

$$\|\mathbf{x}\|_{0,\mathcal{S}} = \sum_{i=1}^{S} \mathbf{1}(\|\mathbf{x}_{\mathcal{S}_i}\|_0 \neq 0),$$

where $\mathbf{x}_{\mathcal{S}_i}$ denotes the vector formed by the entries of $\mathbf{x}$ with index belonging to $\mathcal{S}_i$. Therefore, $\|\boldsymbol{\eta}\|_{0,\mathcal{S}}$ counts the number of subsets modified by an attacker. To guarantee that only one subset is active, an additional constraint can be added to the optimization problem. This leads to the following formulation of the single subset attack for the regression problem:

$$\max_{\boldsymbol{\eta}} \|\mathbf{y} - f(\mathbf{x}+\boldsymbol{\eta})\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_\infty \le \varepsilon\,, \|\boldsymbol{\eta}\|_{0,\mathcal{S}} = 1\,. \tag{16}$$

A similar formulation holds as well for classification problems. The mixed norm $\|.\|_{0,\mathcal{S}}$ in widely used in signal processing and compressed sensing to promoting group sparsity [15].

### A. Single Subset Attack for the Quadratic Problem

As in Section III-A, the approximations $f(\mathbf{x}+\boldsymbol{\eta}) \approx f(\mathbf{x}) + \mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}$ and $\mathbf{y} \approx f(\mathbf{x})$ simplify the problem (16) to

$$\max_{\boldsymbol{\eta}} \|\mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_\infty \le \varepsilon\,,\, \|\boldsymbol{\eta}\|_{0,\mathcal{S}} = 1\,. \tag{17}$$

As it was mentioned above, the problem is NP-hard without the mixed-norm constraint. We try to find an approximate solution to a simpler problem where only the set $\mathcal{S}_s = \{i_s^1,\dots,i_s^Z\}$ is to be modified by the attacker for $s \in [S]$. Finding the perturbation on this set amounts to solving the following problem:

$$\boldsymbol{\eta}_s = \operatorname*{argmax}_{\boldsymbol{\eta}} \|\mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}\|_2^2 \text{ s.t. } \|\boldsymbol{\eta}\|_\infty \le \varepsilon\,, (\boldsymbol{\eta})_i = 0\ \forall i \notin \mathcal{S}_s\,, \tag{18}$$

where $(\boldsymbol{\eta})_i$ denotes the $i$-th entry of $\boldsymbol{\eta}$. As discussed in Section III-A, this problem is NP-hard. Since the maximization of a quadratic bowl over a box constraint lies in the corner points of the feasible set, we have

$$\boldsymbol{\eta}_s = \varepsilon \sum_{z=1}^{Z} \rho_{i_s^z}^* \mathbf{e}_{i_s^z}$$

with $\boldsymbol{\rho}_s^* \triangleq (\rho_{i_s^1}^*,\dots,\rho_{i_s^Z}^*)^\top \in \{-1,+1\}^Z$. The optimization problem can be equivalently formulated as follows:

$$\begin{aligned}\boldsymbol{\rho}_s^* &= \operatorname*{argmax}_{\boldsymbol{\rho}_s\in\{-1,+1\}^Z} \left\|\mathbf{J}_f(\mathbf{x})(\varepsilon\sum_{z=1}^{Z}\rho_{i_s^z}\mathbf{e}_{i_s^z})\right\|_2^2 \\ &= \operatorname*{argmax}_{\boldsymbol{\rho}_s\in\{-1,+1\}^Z} \sum_{z=1}^{Z}\sum_{w=1}^{Z} \rho_{i_s^z}\rho_{i_s^w}\mathbf{J}_{i_s^z}^\top \mathbf{J}_{i_s^w}\,,\end{aligned}$$

for $\boldsymbol{\rho}_s \triangleq (\rho_{i_s^1},\dots,\rho_{i_s^Z})^\top \in \{-1,+1\}^Z$ and $\mathbf{J}_k$ the $k$-th column of $\mathbf{J}_f(\mathbf{x})$. This problem is indeed related to the well

[2]Similar to the so-called $\ell_0$-norm, this is not a proper norm.

known MaxCut problem introduced by [43]. The literature is abound with works on the MaxCut problem, the efficient solutions and their recovery guarantees. A common solution to this problem is a relaxation by a semi-definite programming problem. However, as we discussed semi-definite programming solvers scales badly with the input dimension. Therefore, in the spirit of obtaining fast and scalable approximate solutions, that can later be used to design adversarial perturbations through iterative approximations, we propose to obtain approximate solutions using a greedy approach. To that end, and without loss of generality, let us assume that for a given $\mathcal{S}_s$, the indices $i_s^1, \dots, i_s^Z \in \mathcal{S}_s$ are sorted such that $\|\mathbf{J}_{i_s^1}\|_2 \geq \cdots \geq \|\mathbf{J}_{i_s^Z}\|_2$. An approximate solution for $\rho^*_{i_s^z}$ is calculated in a greedy manner by setting $\rho^*_{i_s^1} = 1$ and recursively calculating

$$\rho^*_{i_s^z} = \operatorname{sign}\left(\left(\sum_{j=1}^{z-1} \rho^*_{i_s^j} \mathbf{J}_{i_s^j}\right)^\top \mathbf{J}_{i_s^z}\right) \quad \forall\, z = 2, \dots, Z\,. \tag{19}$$

As for greedy algorithms, this solution is fast, however, there is no optimality guarantee for it. For the case where $S = 1$ and $S = M$, the expression (19) is an approximate solution for (12) under the $\ell_\infty$-norm constraint on the perturbation (i.e., $p = \infty$).

This method provides an approximate solution to the problem for a given choice of $\mathcal{S}_s$. The solution to (17) can then be obtained by solving the following problem:

$$\boldsymbol{\eta}^* = \boldsymbol{\eta}_{s*}\,, \tag{20}$$
$$\text{with } s^* = \operatorname*{argmax}_s \|\mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}_s\|_2^2 \text{ and } \boldsymbol{\eta}_s = \varepsilon \sum_{z=1}^{Z} \rho^*_{i_s^z} \mathbf{e}_{i_s^z}$$

This is based on naive exhaustive research over the subsets which is tractable only when the number of subsets is small enough.

### *B. Single Subset Attack for the Linear Problem*

Following the steps from Section III-B, we make use of the approximation $L(\mathbf{x}, \boldsymbol{\eta}) \approx L(\mathbf{x}, \widetilde{\boldsymbol{\eta}}) + (\boldsymbol{\eta} - \widetilde{\boldsymbol{\eta}})^\top \nabla L(\mathbf{x}, \widetilde{\boldsymbol{\eta}})$ which leads to the formulation of (16) as a linear programming problem

$$\max_{\boldsymbol{\eta}} \boldsymbol{\eta}^\top \nabla L(\mathbf{x}, \widetilde{\boldsymbol{\eta}}) \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_\infty \leq \varepsilon\,,\, \|\boldsymbol{\eta}\|_{0,\mathcal{S}} = 1\,. \tag{21}$$

In the same manner as Section IV-A, for a given subset $\mathcal{S}_s$ we define $\boldsymbol{\eta}_s$ as in (18). For this linear problem that results in

$$\boldsymbol{\eta}_s = \operatorname*{argmax}_{\boldsymbol{\eta}} \nabla L(\mathbf{x}, \widetilde{\boldsymbol{\eta}})^\top \boldsymbol{\eta} \text{ s.t. } \|\boldsymbol{\eta}\|_\infty \leq \varepsilon\,, (\boldsymbol{\eta})_{i_s^z} = 0\ \forall i_s^z \notin \mathcal{S}_s\,.$$

In contrast to the definition of $\boldsymbol{\eta}_s$ from (18), in this case we have a closed form solution for $\boldsymbol{\eta}_s$ as

$$\boldsymbol{\eta}_s = \varepsilon \sum_{z=1}^{Z} \operatorname{sign}((\nabla L(\mathbf{x}, \widetilde{\boldsymbol{\eta}}))_{i_s^z}) \mathbf{e}_{i_s^z}\,,$$

which implies that $\nabla L(\mathbf{x}, \widetilde{\boldsymbol{\eta}})^\top \boldsymbol{\eta}_s = \sum_{z=1}^{Z} \left|(\nabla L(\mathbf{x}, \widetilde{\boldsymbol{\eta}}))_{i_s^z}\right|$. Therefore, the linear problem for the single subset attack (21) has the closed form solution

$$\boldsymbol{\eta}^* = \boldsymbol{\eta}_{s*}\,,\ \text{with}\ s^* = \operatorname*{argmax}_s \sum_{z=1}^{Z} \left|(\nabla L(\mathbf{x}, \widetilde{\boldsymbol{\eta}}))_{i_s^z}\right| \tag{22}$$

and $\boldsymbol{\eta}_s = \varepsilon \sum_{z=1}^{Z} \operatorname{sign}((\nabla L(\mathbf{x}, \widetilde{\boldsymbol{\eta}}))_{i_s^z}) \mathbf{e}_{i_s^z}$. This results are valid for classification as well when replacing $L$ with $L(\mathbf{x}, \boldsymbol{\eta}) = -(f_{k(\mathbf{x})}(\mathbf{x} + \boldsymbol{\eta}) - \max_{l \neq k(\mathbf{x})} f_l(\mathbf{x} + \boldsymbol{\eta}))$.

## V. Iterative Versions of the Linear Problem

In the previous sections we have formulated several variations of the problem of generating adversarial perturbations. In the same spirit as DeepFool, we make use of the obtained closed form solutions to design adversarial perturbations using iterative approximations. In Algorithm 1 an iterative method based on the linear problem (15) is introduced. This corresponds to a gradient ascent method for maximizing $L(\mathbf{x}, \boldsymbol{\eta})$ with a fixed number of iterations (denoted by $T$) and steps of equal $\ell_p$-norm. Since at every iteration we have that $\|\boldsymbol{\eta}\|_p \leq \varepsilon/T$, increasing $T$ mitigates the error incurred by our linear approximations at expense of increasing computations.

**Algorithm 1** Iterative extension for $\ell_p$ constrained methods.

**input:** $\mathbf{x}$, $f$, $T$, $\varepsilon$, $\widetilde{\varepsilon}_1, \dots, \varepsilon_T$.
**output:** $\boldsymbol{\eta}^*$.
Initialize $\boldsymbol{\eta}_1 \leftarrow \mathbf{0}$.
**for** $t = 1, \dots, T$ **do**
  $\widetilde{\boldsymbol{\eta}}_t \leftarrow \boldsymbol{\eta}_t + \operatorname{random}(\widetilde{\varepsilon}_t)$
  $\boldsymbol{\eta}_t^* \leftarrow \operatorname{argmax}_{\boldsymbol{\eta}} \boldsymbol{\eta}^\top \nabla L(\mathbf{x}, \widetilde{\boldsymbol{\eta}}_t)$ s.t. $\|\boldsymbol{\eta}\|_p \leq \varepsilon/T$ (Table I)
  $\boldsymbol{\eta}_{t+1} \leftarrow \boldsymbol{\eta}_t + \boldsymbol{\eta}_t^*$
**end for**
**return:** $\boldsymbol{\eta}^* \leftarrow \boldsymbol{\eta}_T$

While generalizing the results for (15) into a gradient ascent method is trivial, the same is not true for the quadratic problem (12). The main reason for this is that, using the approximation $\mathbf{y} \approx f(\mathbf{x})$, we were able to simplify (11) into (12) since $\mathbf{y} - f(\mathbf{x}) \approx \mathbf{0}$. For an iterative version of this solution we must successively approximate $f(\cdot)$ around different points $\widetilde{\mathbf{x}}$, which leads to $\mathbf{y} - f(\widetilde{\mathbf{x}}) \neq \mathbf{0}$ even if $\mathbf{y} = f(\mathbf{x})$. We leave the task of investigating alternatives for designing iterative methods with the results for (12) for future works, and in Section VI show that the non-iterative solutions for this method are still top performing.

Finally, replacing line 5 of Algorithm 1 with

$$\boldsymbol{\eta}_t^* \leftarrow \operatorname*{argmax}_{\boldsymbol{\eta}} \boldsymbol{\eta}^\top \nabla L(\mathbf{x}, \widetilde{\boldsymbol{\eta}}_t) \text{ s.t. } \|\boldsymbol{\eta}\|_p \leq \varepsilon\,, \|\boldsymbol{\eta}\|_{0,\mathcal{S}} = 1$$

leads to a multiple subset attack, since we modify the values of one subset at every iteration. At every iteration, we may exclude the previously modified subsets from $\mathcal{S}$ in order to ensure that a new subset is modified.

## VI. Experiments

In this section, the proposed methods are used to fool neural networks in classification and regression problems. But first, we summarize the presented algorithms and the relation with other existing attacks. The proposed attacks are summarized in Table I. These attacks rely on perturbation analysis of learning algorithms and can be used for classification and regression tasks. The approach we chose to derive these algorithms is

TABLE I
SUMMARY OF THE OBTAINED CLOSED-FORM SOLUTIONS FOR ADVERSARIAL PERTURBATIONS

| Type of Attack | Problem | Adversarial Perturbation | Exact Solution | Application |
|---|---|---|---|---|
| $\ell_2$-constrained attack | (15) | (13) | Yes | Regression |
| $\ell_2$-constrained attack | (12) | (5) | Yes | Classification/Regression |
| $\ell_\infty$-constrained attack | (15) | (20) | No | Regression |
| $\ell_\infty$-constrained attack | (12) | (6) | Yes | Classification/Regression |
| Single-subset attacks | (17) | (20) | No | Regression |
| Single-subset attacks | (21) | (22) | Yes | Classification/Regression |

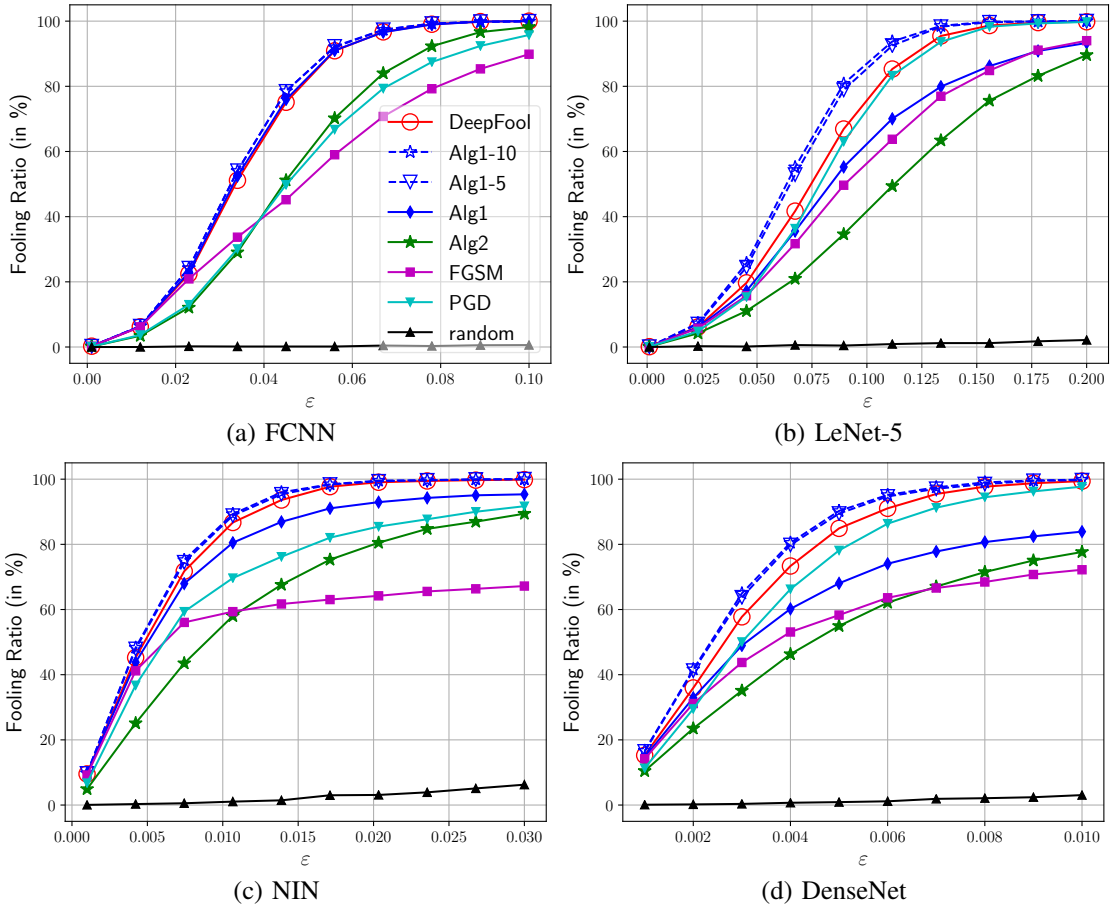



Fig. 1. (a) and (b): Fooling ratio of the adversarial samples for different values of $\varepsilon$ on the MNIST test dataset. (c) and (d): Fooling ratio of the adversarial samples for different values of $\varepsilon$ on the CIFAR-10 test datasets.

TABLE II
SUMMARY OF EXISTING ATTACKS IN CLASSIFICATION OBTAINED FROM PRESENTED ALGORITHMS AGP.II

| Attack | Design Algorithm | Iterative | Dithering |
|---|---|---|---|
| FGSM [12] | (4) with cross-entropy loss | × | × |
| R-FGSM [22] | (4) with cross-entropy loss | × | ✓ |
| BIM [17] | (4) with cross-entropy loss | ✓ | × |
| PGD [16] | (4) with cross-entropy loss | ✓ | ✓ |
| DeepFool [14] | (7) with $p = 2$ | ✓ | × |
| Ensemble [22] | (4) with combined cross-entropies | × | ✓ |
| Targeted | (4) with (23) loss | ✓ | ✓ |
| Algorithm 1 | (4) | ✓ | ✓ |

capable of rendering other existing methods by adjusting the choice of $L(\mathbf{x}, \cdot)$ and other design parameters. This point has been discussed in Table II. In that table we also include the black-box ensemble attack from [22] and targeted attacks [13], [21], [26], [36], [44]. Consider first the ensemble attack of [22]. A black-box attack in nature, this attack does not have access to the target neural network function and uses another, potentially similar to the neural network function $f$, hoping that the obtained adversarial example transfers to the unknown network. After choosing multiple similar neural networks, the attack adopts the loss function as the averaged sum of cross entropies of these networks. Once the loss function is fixed, the algorithm, similar to above methods, uses a combination of perturbation analysis with an optimization problem of type (4) or (7) to solve the problem. Targeted attacks are used when the goal is to generate adversarial examples that are classified by target system as belonging to some given target class $l \in [K]$.

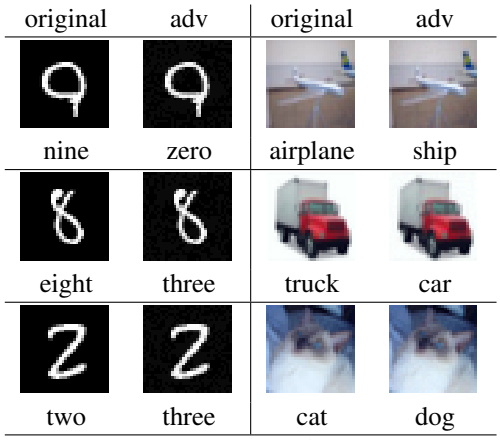



Fig. 2. Examples of correctly classified images that are misclassified when adversarial noise is added using Algorithm 1 with $T = 5$.

That corresponds to fixing the loss function to

$$L(\mathbf{x}, \boldsymbol{\eta}) = f_{k(\mathbf{x})}(\mathbf{x} + \boldsymbol{\eta}) - f_l(\mathbf{x} + \boldsymbol{\eta}). \tag{23}$$

After that, the process of finding adversarial examples leverages the same techniques, i.e., perturbation analysis and convex optimization, discussed in this paper. This is, however, only a side feature of our approach. Our proposed attacks are still of independent interest.

The goal of this section is twofold. First, we would like to examine the performance of the newly proposed attack in classification tasks, thereby showing the strength of our proposed method. Secondly we generate adversarial perturbations for various regression tasks which only received small attention in the literature. For this purpose we use the MNIST [45], CIFAR-10 [46], STL-10 and PASCAL VOC 2012 datasets.

### A. Classification

As discussed in Section II, the appropriate loss function $L(\mathbf{x}, \boldsymbol{\eta})$ for image classification tasks that should be used in (4) is given by (AGP). For this problem, $\|\boldsymbol{\eta}\|_\infty \leq \varepsilon$ is a common constraint that models the undetectability, for sufficiently small $\varepsilon$, of adversarial noise by an observer. However solving (4) involves finding the function $L(\mathbf{x}, \mathbf{0})$ which is defined as the minimum of $K - 1$ functions with $K$ being the number of different classes. In large problems, this may significantly increase the computations required to fool one image. Therefore, we include a simplified version of this algorithm in our simulations. The non-iterative methods might not guarantee the fooling of the underlying network but on the other hand, the iterative methods might suffer from convergence problems.

To benchmark the proposed adversarial algorithms, we consider following methods tested on the aforementioned datasets:

- **Algorithm 1**: This algorithm solves (4) with $L(\mathbf{x}, \cdot)$ given by (AGP). Note that, for evaluating $L$ at a given $\mathbf{x}$ one must search over all $l \neq k(\mathbf{x})$. This can be computationally expensive when the number of possible classes (i.e., the number of possible values for $l$) is large. The $\ell_\infty$-norm is chosen for the constraint. Moreover, an example of adversarial images obtained using this algorithm is shown in Figure 3.
- **Algorithm 1**-$T$: This is the iterative version of Algorithm 1 with $T$ iterations. The adversarial perturbation is the sum of $T$ perturbation vectors with $\ell_\infty$-norm of $\varepsilon/T$ computed through $T$ successive approximations.
- **Algorithm 2**: This algorithm approximates (AGP) with $L(\mathbf{x}, \boldsymbol{\eta}) \approx f_{k(\mathbf{x})}(\mathbf{x} + \boldsymbol{\eta})$, thus reducing the computation of $L(\mathbf{x})$ when the number of classes is large. Note that we cannot use $L(\mathbf{x}, \boldsymbol{\eta}) < 0$ to guarantee that we have fooled the network. Nevertheless, the lower the value of $L(\mathbf{x}, \boldsymbol{\eta})$ the most likely it is that the network has been fooled. The same reasoning is valid for the FGSM algorithm.

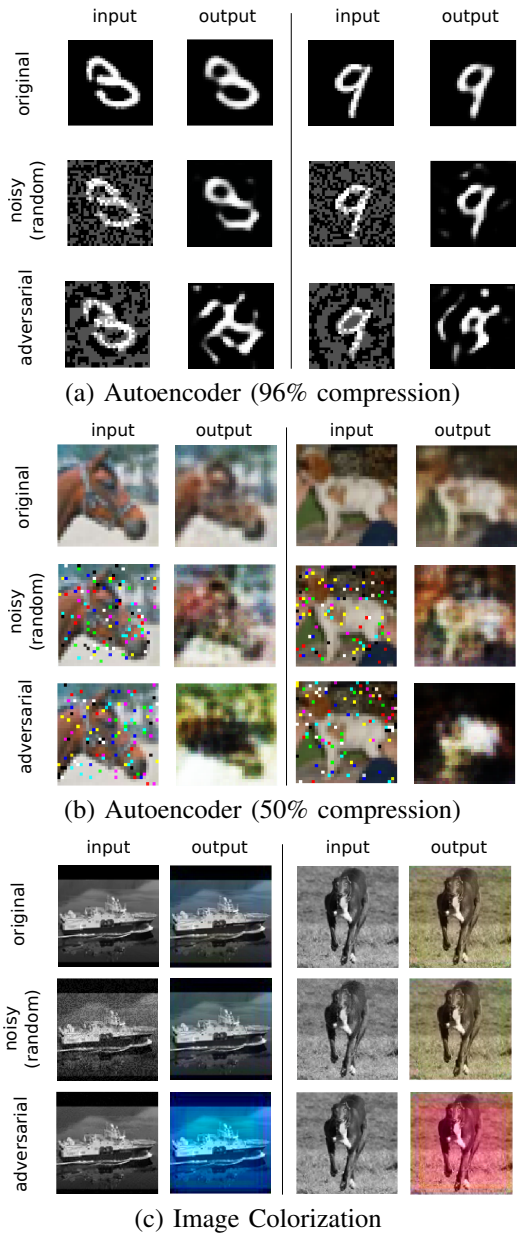



Fig. 3. Adversarial examples for (a): MNIST autoencoder obtained using *quadratic*-$\ell_\infty$ with input PSNR = 17dB, (b): CIFAR-10 autoencoder obtained using *linear-pixel*-100 and $\varepsilon = 1$, (c): STL-10 colorization network obtained using *linear*-$\ell_\infty$-20 with input PSNR = 25dB.

TABLE III
ROBUSTNESS MEASURES FOR DIFFERENT CLASSIFIERS

| | Test error | $\widehat{\rho}_1(f)$ [14] | $\widehat{\rho}_2(f)$ (ours) | fooled >99% |
|---|---|---|---|---|
| FCNN (MNIST) | 1.7% | 0.036 | 0.034 | $\varepsilon$ =0.076 |
| LeNet-5 (MNIST) | 0.9% | **0.077** | **0.061** | $\varepsilon$ =**0.164** |
| NIN (CIFAR-10) | 13.8% | **0.012** | **0.004** | $\varepsilon$ =**0.018** |
| DenseNet (CIFAR-10) | 5.2% | 0.006 | 0.002 | $\varepsilon$ =0.010 |

- **FGSM**: This well-known method was proposed by [12] where $L(\mathbf{x}, \boldsymbol{\eta})$ is replaced by the negative of the training loss for the input $\mathbf{x}+\boldsymbol{\eta}$. Usually the cross-entropy loss is used for this purpose. With the newly replaced function, (4) is solved for $p = \infty$.
- **PGD**: This method, given in [16], is the iterative version of FGSM ($T > 1$) with $\widetilde{\varepsilon}_1 = \varepsilon$ and $\widetilde{\varepsilon}_t = 0$ for all $t > 1$. It constitutes one of the state-of-the-art attacks in the literature.
- **DeepFool**: This method was proposed in [14] and makes use of iterative approximations. Every iteration of DeepFool can be written within our approach by replacing $L$ by

$$L(\mathbf{x}, \boldsymbol{\eta}) = f_{k(\mathbf{x})}(\mathbf{x}+\boldsymbol{\eta}) - f_{\widehat{l}}(\mathbf{x}+\boldsymbol{\eta}), \quad \text{where}$$
$$\widehat{l} = \underset{l \neq k(\mathbf{x})}{\operatorname{argmin}} \left\{ \frac{|f_{k(\mathbf{x})}(\mathbf{x}) - f_l(\mathbf{x})|}{\|\nabla f_{k(\mathbf{x})}(\mathbf{x}) - \nabla f_l(\mathbf{x})\|_q} \right\}.$$

  The adversarial perturbations are computed using $p = \infty$, thus $q = 1$, with a maximum of 50 iterations. These parameters were taken from [14]. Note that $\widehat{l}$ is chosen to minimize the robustness $\widehat{\rho}_1(f)$ for $\mathcal{D} = \{\mathbf{x}\}$.
- **Random**: For benchmarking purposes, we also consider random perturbations with independent Bernoulli distributed entries with $\mathbb{P}(\varepsilon) = \mathbb{P}(-\varepsilon) = \frac{1}{2}$. This helps to demarcate the essential difference of adversarial and random perturbations.

The above methods are tested on the following deep neural network architectures:

- **MNIST** : A fully connected network (FCNN) with two hidden layers of size 150 and 100 respectively, as well as the LeNet-5 architecture [47].
- **CIFAR-10** : The Network In Network (NIN) architecture [48], and a 40 layer DenseNet [49].

As a performance measure, we use the *fooling ratio* defined in [14] as the percentage of correctly classified images that are missclassified when adversarial perturbations are applied. Of course, the fooling ratio depends on the constraint on the norm of adversarial examples. Therefore, in Figure 1 we observe the fooling ratio for different values of $\varepsilon$ on the aforementioned neural networks. As expected, the increased computational complexity of iterative methods such as DeepFool and Algorithm 1-$T$ translates into increased performance with respect to non-iterative methods. Nevertheless, as shown in Figures 1(a) and (c), the performance gap between iterative and non-iterative algorithms is not always significant. Such phenomenon is present when classifiers are highly linear in the vicinity around the inputs, since non-iterative would already yield accurate approximations, thus leaving not much left to gain by refining these approximations using iterative procedures. For the case of iterative algorithms, the proposed Algorithm 1-$T$ outperforms DeepFool and PGD. The same holds true for Algorithm 1 with respect to other non-iterative methods such as FGSM, while Algorithm 2 obtains top performance with respect to FGSM. However, note that adversarial training using PGD is the state-of-the-art defense against adversarial examples, thus PGD may still be a better choice than Algorithm 1-$T$ for adversarial training. Some adversarial examples obtained using Algorithm 1-$T$ are shown in Figure 2. Finally, we measure the robustness of different networks using $\widehat{\rho}_1(f)$ and $\widehat{\rho}_2(f)$, with $p = \infty$. We also include the minimum $\varepsilon$, such that DeepFool obtains a fooling ratio greater than 99%, as a performance measure as well. These results are summarized in Table III, where we obtain coherent results between the 3 measures.

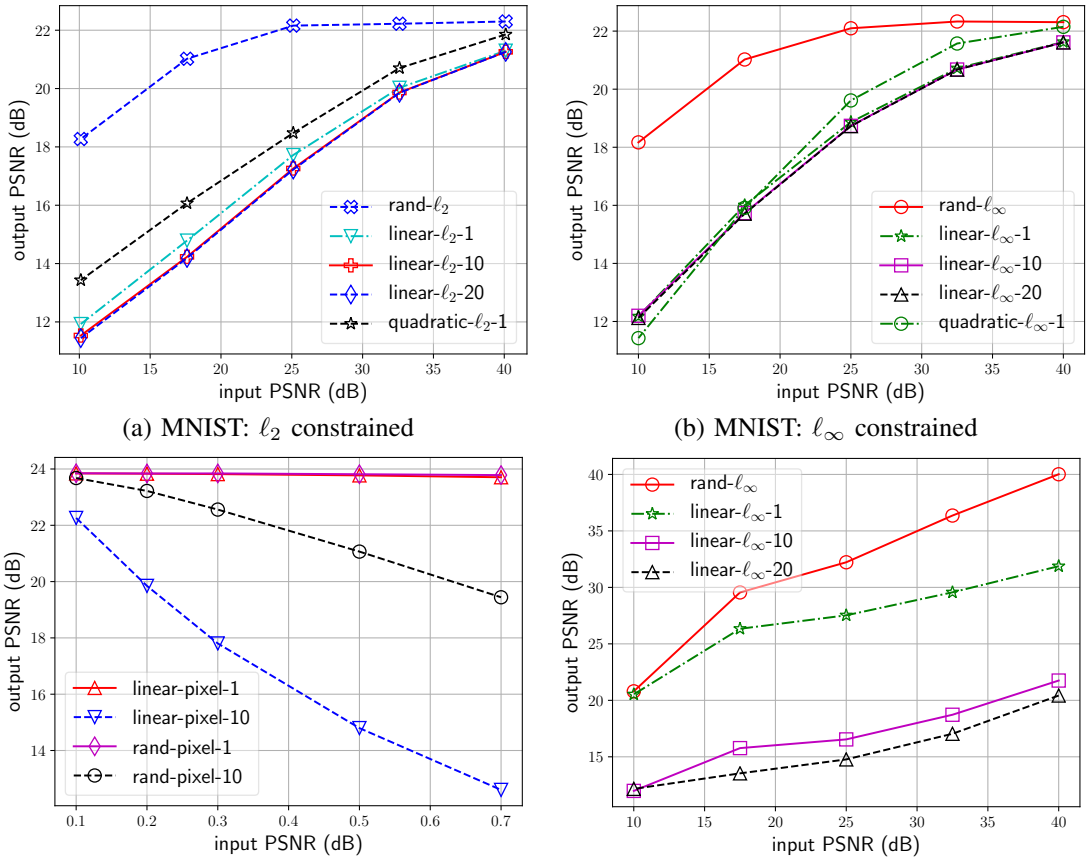
### *B. Regression*

For the sake of clarity we use the following notation to distinguish between the different regression methods used in this section:

- **quadratic-$\ell_p$**: This algorithm computes adversarial perturbations by solving the quadratic problem (12) under the $\ell_p$-norm constraint.
- **linear-$\ell_p$-$T$**: Similarly, this attack refers to Algorithm 1 with $T$ iterations and the $\ell_p$-norm constraint.
- **linear-pixel-$T$**: Since the experiments carried out in this section are exclusively image based, we use this notation to refer to the multiple subset attack with $\|\boldsymbol{\eta}\|_{0,\mathcal{S}} = T$.
- **random-$\ell_p$**: Similarly to [14], we show the validity of our attacks by comparing their performance against appropriate types of random noise. For $p = 2$, the random perturbation is computed as $\boldsymbol{\eta} = \varepsilon\, \mathbf{w}/\|\mathbf{w}\|_2$, where the entries of $\mathbf{w}$ are independently drawn from a Gaussian distribution. For $p = \infty$ the random perturbation $\boldsymbol{\eta}$ has independent Bernoulli distributed entries with parameter $1/2$.
- **random-pixel-$T$**: Random attacks are considered as well for multiple subset attacks with $p = \infty$. The random perturbations are added to only $T$ randomly chosen pixels, while the other pixels are untouched.

The random attacks are used for benchmarking purposes. The algorithms behind our proposed attacks, i.e., the first three attacks, are summarized in Table I. Since the aim of the proposed attacks is to maximize the MSE of the target system, we use the Peak Signal to Noise Ratio, which is a common measure for image quality and is defined as $\mathrm{PSNR} = (\text{maximum pixel value})^2/\mathrm{MSE}$, as the performance metric. As an example, assume $\|\mathbf{x}\|_\infty \leq 1$ and $\|\boldsymbol{\eta}\|_\infty \leq \varepsilon$, then this performance metric simplifies to $\mathrm{PSNR} = 1/n\varepsilon^2$.

For our experiments we use the MNIST, CIFAR-10 and STL-10 and PASCAL VOC 2012 datasets. A different neural network is trained for each of these datasets. As in [30], we also consider autoencoders. For MNIST and CIFAR-10 we have trained fully connected autoencoders with 96% and 50% compression rates respectively. Next, we train the image colorization architecture from [50] for the STL-10 dataset. Finally, we use the YOLO architecture for object detection from [51]. The convolutional




(a) MNIST: $\ell_2$ constrained

(b) MNIST: $\ell_\infty$ constrained

(c) CIFAR-10: Multiple pixel attack

(d) STL-10 (colorization): $\ell_\infty$ constrained

Fig. 4. Output PSNR for (a): MNIST autoencoder under $\ell_2$-norm constraint, (b): MNIST autoencoder under $\ell_\infty$-norm constraint, (c): CIFAR-10 autoencoder under multiple pixel attacks, (d): STL-10 colorization network under $\ell_\infty$-norm constraint.

layers of the proposed network are trained on the ImageNet dataset with 1000 classes. The experiments are run on the PASCAL VOC 2012 dataset [52], which is a common dataset for object detection tasks.

Different example images obtained from applying the proposed methods on these networks are shown in Figure 3. For instance, in Figure 3(a) we observe that the autoencoder trained on MNIST is able to denoise random perturbation correctly but fails to do so with adversarial perturbations obtained using the *quadratic*-$\ell_\infty$ method. Similarly, in Figure 3(b), the *random-pixel*-100 algorithm distorts the output significantly more than its random counterpart. These two experiments align with the observation of [30] that autoencoders tend to be more robust to adversarial attacks than deep neural networks used for classification. The deep neural network trained for colorization is highly sensitive to adversarial perturbations as illustrated in Figure 3(c), where the original and adversarial images are nearly identical.

While the results shown in Figure 3 are for some particular images, in Figure 4 we measure the performance of different adversarial attacks using the average output PSNR over 20 randomly selected images from the corresponding datasets. In Figures 4(a) and 4(b) we observe how computing adversarial perturbations through successive linearizations improves the performance. This behavior is more pronounced in Figure 4(d), where iterative linearizations are responsible for more than 10 dB of output PSNR reduction. Note that, in Figures 4(a) and 4(b) the non-iterative *quadratic*-$\ell_p$ algorithm performs competitively, even when compared to iterative methods. In Figure 4(c) we observe that the autoencoder trained on CIFAR-10 is robust to single pixel attacks. However, an important degradation of the systems performance, with respect to random noise, can be obtained through adversarial perturbations in the 100 pixels attack ($\approx 9.7\%$ of the total number of pixels). Moreover, in Figure 4(d), we can clearly observe the instability of the image colorization network to adversarial attacks. These experiments show that, even though some regression problems can tolerate adversarial noise like in Figures 4(a) and 4(b), others (such as the ones from Figures 4(c) and (d)) are highly unstable.

Finally, object detection tasks can be treated as regression problems, since the predicted location and size of the detected object can be approximated as continous variables. Therefore, we can observe in Figure 5(a) how the performance of YOLO at detecting objects is serverly degraded when adding adversarial perturbations. The loss function used in this experiment is the same used in [51], which accounts for the correct labeling of the detected objects as well as the correct placement of the boxes surrounding them. Some concrete example images from the PASCAL VOC 2012 dataset are shown in Figure 5(b).

These experiments show that, even though autoencoders are somehow robust to adversarial noise, this may not be true for deep neural networks in other regression problems.

## VII. Conclusion

The perturbation analysis of different learning algorithms underlies many attacks. In this work, the generation of adversarial examples is pursued by applying perturbation analysis to a fairly general problem and is formulated as a convex program. The other techniques like iterative methods and randomization of instances can be additionally considered. We used this generic approach to propose new attacks for classification and regression problems under various desirable constraints. This includes in particular single-pixel and single-subset attacks. Through multiple examples, regression problems are shown to be equally vulnerable to adversarial perturbations. Our new attacks on classification functions are benchmarked through empirical simulations of the fooling ratio against the well-known FGSM, DeepFool, and PGD methods. For regression tasks, three use cases are considered, namely, autoencoders, images colorization and object detection algorithms.

For classification tasks, the adversarial vulnerability is directly related to the properties of classification margin. Regression algorithms, however, do not dispose such a notion. It is an interesting research question to investigate the vulnerabilities of these algorithms.

## References


[1] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups," *IEEE Signal Processing Magazine*, pp. 82–97, 2012.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1137–1149, 2017.

[6] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2014.

[7] A. Fawzi, S.-M. Moosavi-Dezfooli, and P. Frossard, "Robustness of classifiers: From adversarial to random noise," in *Neural Information Processing Systems (NIPS)*, 2016, pp. 1632–1640.

[8] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, "The security of machine learning," *Machine Learning*, pp. 121–148, 2010.

[9] N. Akhtar and A. Mian, "Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey," *IEEE Access*, pp. 14 410–14 430, 2018.

[10] B. Wang, J. Gao, and Y. Qi, "A Theoretical Framework for Robustness of (Deep) Classifiers against Adversarial Examples," in *International Conference on Learning Representations (ICLR)*, 2017.

[11] A. Fawzi, O. Fawzi, and P. Frossard, "Fundamental limits on adversarial robustness," in *ICML Workshop on Deep Learning*, 2015.

[12] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," in *International Conference on Learning Representations (ICLR)*, 2015.

[13] S. Sarkar, A. Bansal, U. Mahbub, and R. Chellappa, "Upset and angri: Breaking high performance image classifiers," *arXiv preprint arXiv:1707.01159*, 2017.

[14] S. M. Moosavi Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[15] N. Rao, B. Recht, and R. Nowak, "Universal measurement bounds for structured sparse signal recovery," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012, pp. 942–950.

[16] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards Deep Learning Models Resistant to Adversarial Attacks," in *International Conference on Learning Representations (ICLR)*, 2018.

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

[18] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples," in *International Conference on Machine Learning (ICML)*, 2018, pp. 274–283.

[19] A. Athalye and N. Carlini, "On the robustness of the CVPR 2018 white-box adversarial example defenses," *arXiv preprint arXiv:1804.03286*, 2018.

[20] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, 2019.

[21] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2016, pp. 372–387.

[22] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble Adversarial Training: Attacks and Defenses," in *International Conference on Learning Representations (ICLR)*, 2018.

[23] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 86–94.

[24] J. H. Metzen, M. C. Kumar, T. Brox, and V. Fischer, "Universal adversarial perturbations against semantic image segmentation," pp. 2774–2783, 2017.

[25] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille, "Adversarial examples for semantic segmentation and object detection," in *IEEE International Conference on Computer Vision (CVPR)*, 2017.

[26] M. Cisse, Y. Adi, N. Neverova, and J. Keshet, "Houdini: Fooling deep structured prediction models," *arXiv preprint arXiv:1707.05373*, 2017.

[27] N. Papernot, P. McDaniel, A. Swami, and R. Harang, "Crafting adversarial input sequences for recurrent neural networks," in *IEEE Military Communications Conference (MILCOM)*, IEEE, 2016, pp. 49–54.

[28] Y.-C. Lin, Z.-W. Hong, Y.-H. Liao, M.-L. Shih, M.-Y. Liu, and M. Sun, "Tactics of adversarial attack on deep reinforcement learning agents," in *International Joint Conference on Artificial Intelligence (IJCAI)*, AAAI Press, 2017, pp. 3756–3762.

[29] J. Kos, I. Fischer, and D. Song, "Adversarial Examples for Generative Models," in *IEEE Security and Privacy Workshops (SPW)*, 2018, pp. 36–42.

[30] P. Tabacof, J. Tavares, and E. Valle, "Adversarial images for variational autoencoders," *arXiv preprint arXiv:1612.00155*, 2016.

[31] T. Tanay and L. Griffin, "A boundary tilting persepective on the phenomenon of adversarial examples," *arXiv preprint arXiv:1608.07690*, 2016.

[32] A. Fawzi, S. M. Moosavi-Dezfooli, and P. Frossard, "The Robustness of Deep Networks: A Geometrical Perspective," *IEEE Signal Processing Magazine*, pp. 50–62, 2017.

[33] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified Defenses against Adversarial Examples," in *International Conference on Learning Representations (ICLR)*, 2018.

[34] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," in *International Conference on Learning Representations (ICLR)*, 2019.

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

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

[37] E. R. Balda, A. Behboodi, and R. Mathar, "On generation of adversarial examples using convex programming," in *52-th Asilomar Conference*

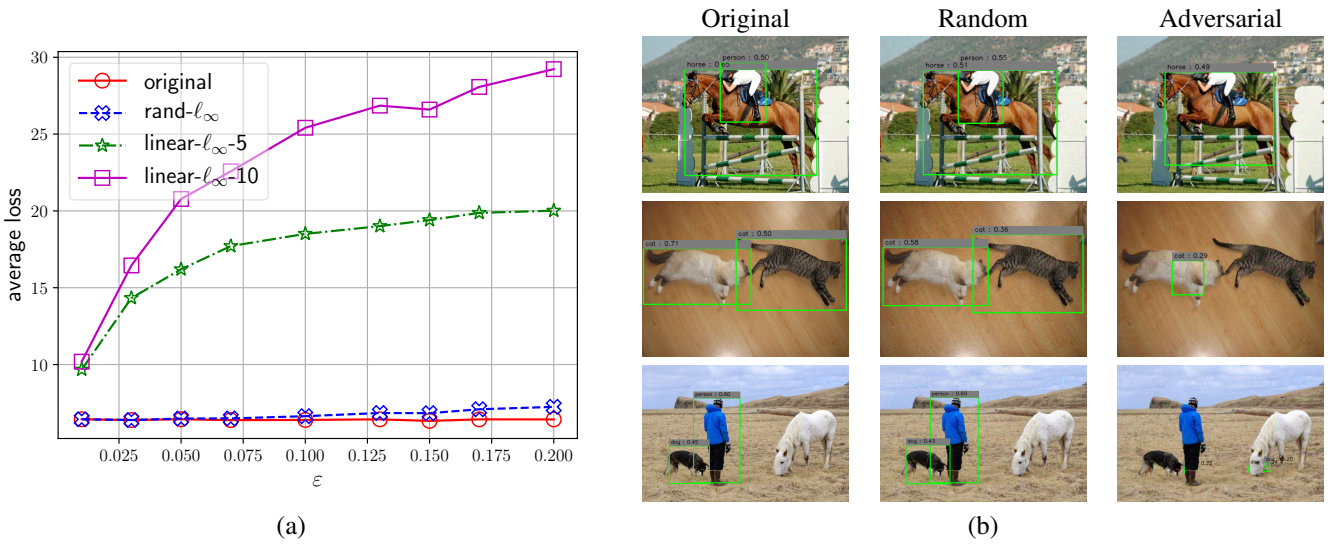



Fig. 5. The YOLO architecture under $\ell_\infty$-norm constraint in the PASCAL VOC 2012 dataset. (a) Output loss, defined as in [51]. (b) Adversarial examples obtained using *linear-*$\ell_\infty$-10.


*on Signals, Systems, and Computers*, Pacific Grove, California, USA, 2018, pp. 1–6.

[38] M. Hein and M. Andriushchenko, "Formal guarantees on the robustness of a classifier against adversarial manipulation," in *Neural Information Processing Systems (NIPS)*, 2017.

[39] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge Univ. Press, 2013.

[40] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing*. New York, NY: Springer New York, 2013.

[41] J. Rohn, "Computing the norm $\|A\|_{\infty,1}$ is NP-hard," *Linear and Multilinear Algebra*, pp. 195–204, 2000.

[42] D. Hartman and M. Hladík, "Tight Bounds on the Radius of Nonsingularity," in *Scientific Computing, Computer Arithmetic, and Validated Numerics*, Springer, 2015, pp. 109–115.

[43] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *Journal of the ACM (JACM)*, pp. 1115–1145, 1995.

[44] S. Baluja and I. Fischer, "Adversarial transformation networks: Learning to generate adversarial examples," *arXiv preprint arXiv:1703.09387*, 2017.

[45] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2010.

[46] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[47] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," in *Shape, contour and grouping in computer vision*, Springer, 1999, pp. 319–345.

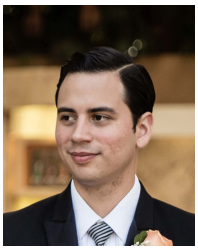
[48] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[49] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, p. 3.

[50] F. Baldassarre, D. G. Morín, and L. Rodés-Guirao, "Deep koalarization: Image colorization using cnns and inception-resnet-v2," *arXiv preprint arXiv:1712.03400*, 2017.

[51] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[52] M. Everingham and J. Winn, *The pascal visual object classes challenge 2012 (voc2012) development kit*, 2011.

**Emilio Balda** received the M.Sc. degree in communications and signal processing from the Ilmenau University of Technology, Ilmenau, Germany, in 2017. Since 2017, he has been a Research Assistant with the Institute for Theoretical Information Technology, RWTH Aachen University. His research interests are machine learning, optimization and signal processing.

**Arash Behboodi** received his B.Sc. and M.Sc. degrees in electrical engineering from Sharif University of Technology, Tehran, Iran in 2005 and 2007 and Ph.D. degree from École Supérieure délectricité (now CentraleSupélec), Gif-sur-Yvette, France in 2012. He is currently a deep learning researcher in Qualcomm AI research. His research interests are information theory, compressed sensing and machine learning.

**Rudolf Mathar** received the Ph.D. degree from RWTH Aachen University in 1981. Previous positions include lecturer positions with Augsburg University and the European Business School. In 1989, he joined the Faculty of Natural Sciences, RWTH Aachen University. He has held the International IBM Chair in Computer Science with Brussels Free University in 1999. In 2004, he was appointed as the Head of the Institute for Theoretical Information Technology with the Faculty of Electrical Engineering and Information Technology, RWTH Aachen University. Since 1994, he has held numerous visiting professor positions with The University of Melbourne, the University of Canterbury, Christchurch, New Zealand, Johns Hopkins University, Baltimore, MD, USA, and others. From 2011 to 2014, he served as the Dean of the Faculty of Electrical and Engineering and Information Technology. Since 2014, he has been serving as a Prorector for research and structure with RWTH Aachen University. In 2002, he was a recipient of the prestigious Vodafone Innovation Award, and in 2010, he was an elected member of the NRW Academy of Sciences and Arts. He is a co-founder of three spin-off enterprises. In 2012, he was an elected speaker of the Board of Deans with RWTH Aachen University. His research interests include information theory, mobile communication systems, particularly optimization, resource allocation, and access control.