Adversarial Examples for Deep Neural Networks in Regression Problems

Anonymous Author(s)
Affiliation
Address
email

Abstract

Deep neural networks have demonstrated remarkable performance in visual recognition tasks. However, it has been observed that adding specifically designed adversarial noise to the inputs of these architectures leads to erroneous classification, with high confidence in the prediction. In this work we consider the regression instead of classification. Through recursive first order approximations, we compute adversarial examples as linear combinations of closed form solutions from convex problems. Using this method, we are able to obtain small perturbations that produce high variations on the estimates of deep neural networks. We illustrate this phenomenon using experiments with images, such as lossy compression and image colorization. [1]

1 Introduction

Deep neural networks have achieved outstanding performance in computer vision, language processing and other tasks. Despite their popularity and success, Szegedy et al. (2013) discovered that when classifying images, these architectures suffer from instability against maliciously designed perturbations on the input data. The perturbed inputs, known as adversarial examples, are visually indistinguishable from the original images by a human observer despite being able to change the outcome of the classification. In addition, the target neural networks tend to have high confidence in their wrongful classification of adversarial examples. A common approach to generating adversarial examples that are indistinguishable from the original images is to constrain the ℓ_p norm of the adversarial perturbation for $p > 1$. For instance, Szegedy et al. (2013) proposed a method for finding minimal ℓ_2 perturbations by solving a box-constrained non-convex optimization problem. This gave rise to various attacks such as the Fast Gradient Sign Method (FGSM) Goodfellow et al. (2014), which obtains perturbations with upper-bounded ℓ_∞ norm, and the closely related method from Miyato et al. (2017). More sophisticated attacks like the DeepFool algorithm Moosavi Dezfooli et al. (2016) obtain minimum ℓ_p norm perturbations through successive linearization of the neural network's function, while Carlini and Wagner (2017) proposed three attacks to bypass defensive distillation of the adversarial perturbations Papernot et al. (2016c). Another popular way of generating adversarial examples is by contraining the ℓ_0 norm of the perturbation. These types of attacks are known as single pixel attacks Su et al. (2017) and multiple pixel attacks Papernot et al. (2016a). While most of these attacks assume some knownledge of the target system parameters, Sarkar et al. (2017) developed a black box attack where no information about the target neural network is required. Moreover, Moosavi-Dezfooli et al. (2017) showed the existence of universal adversarial perturbations that are independent from the system and the target input.

[1] In the spirit of encouraging reproducible research, the tensorflow implementations used in this paper have been made available at `https://github.com/adversarialregerssion/nips2018`.

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.

The aforementioned methods were designed for attacking image classification and recognition tasks. 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, Metzen et al. (2017) constructed an attack on image segmentation, while Xie et al. (2017) designed attacks for object detection. The Houdini attack Cisse et al. (2017) aims at distorting speech recognition systems. Moreover, Papernot et al. (2016b) taylored an attack for recurrent neural networks, and Lin et al. (2017) for reinforcement learning. Adversarial examples exist for probabilistic methods as well. For instance, Kos et al. (2017) showed the existence of adversarial examples for generative models.

For regression problems, Tabacof et al. (2016) designed an attack that specifically targets autoencoders. In that work, the perturbation is designed such that the pixel values of the autoencoder's output is as close as possible to some target image, which is distinct from the original input. In this work we extend this idea by formulating the problem of designing adversarial perturbations for regression problems in general. The objective of our attack is to distort the output of system as much as possible without a target in mind, which distinguishes this attack from the work of Tabacof et al. (2016). To that end we design adversarial perturbations, with bounded ℓ_p norms, that maximize the Mean Squared Error (MSE) at the output of the system. Note that this objective is different from what its used for attacking classifiers, since maximizing the MSE of classifier is not equivalent maximizing its classification error. We make use of the first order perturbation analysis of the target system to derive convex optimization problems. The quadratic nature of the objective allows us to derive quadratic and linear programming formulations, with closed form solutions, for generating adversarial examples. In addition, we formulate single and multiple pixel attacks through slight modifications of the presented methods. In the same spirit as Moosavi Dezfooli et al. (2016), we use these results to design adversarial noise through iterative approximations. Even though the aim of this work is to expose the existence of adversarial examples for general regression problems, we use image based systems to illustrate our results. Through empirical simulations we show the existence of adversarial examples, and the validity of the obtained methods, for autoencoders and image colorization. In these experiments, the tested neural networks are sensitive to adversarial noise despite being robust to random noise.

This paper is organized as follows. In Section 2, we formulate the problem of generating adversarial examples for regression systems in two different ways, as a quadratic and a linear programming problem. Then, in Section 3, we modify these problems to derive single pixel attacks, which we denote as single subset attacks. These results are extended to their corresponding iterative versions in Section 4. Finally, the validity of the obtained methods is demonstrated in Section 5 through empirical simulations.

2 System Model

In classical statistical learning theory, a regression problem is 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 $\mathbb{E}_P(L(f(\mathbf{x}), \mathbf{y}))$, where $L : \mathbb{R}^M \times \mathbb{R}^K \to \mathbb{R}$ is a function that measures the similarity between $f(\mathbf{x})$ and $\mathbf{y}$. While logarithmic losses are popular in classification problems, the squared loss $L(f(\mathbf{x}), \mathbf{y}) = \|f(\mathbf{x}) - \mathbf{y}\|_2^2$ is mostly used for the general regression setting. For the sake of notation, given $\mathbf{y}$ and f, let us define $L(\mathbf{x}) = L(f(\mathbf{x}), \mathbf{y})$. Furthermore, for a given f, $\mathbf{x}$ and $\mathbf{y}$, it is the aim of an attacker to find an additive perturbation vector $\boldsymbol{\eta}$ that is "imperceptible" to the owners of the target system, while maximizing the loss of the perturbed input $L(\mathbf{x} + \boldsymbol{\eta})$ as

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

A common practice to model the imperceptibility of the perturbation is to upper bound its ℓ_p-norm (for some $p \geq 1$) by a small value $\varepsilon \in \mathbb{R}_+$. This leads to the following formulation of the problem

$$\max_{\boldsymbol{\eta}} L(\mathbf{x} + \boldsymbol{\eta}) \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_p \leq \varepsilon\,, \text{ with} \quad L(\mathbf{x} + \boldsymbol{\eta}) = \|\mathbf{y} - f(\mathbf{x} + \boldsymbol{\eta})\|_2^2\,. \tag{1}$$

Note that, while we assume f to be known, $\mathbf{y}$ is generally unknown. For classification this problem is generally tackled by assuming that f does a good job at predicting $\mathbf{y}$, thus the approximation $\mathbf{y} \approx f(\mathbf{x})$ can be made. Nevertheless, there are cases where by the nature of the problem $\mathbf{y}$ is known. For instance, when f is an encoder-decoder pair such as an autoencoder we have that $\mathbf{y} = \mathbf{x}$. For the sake of generality, we use the formulation in (1) and discuss the implications of applying the approximation $\mathbf{y} \approx f(\mathbf{x})$ in later sections.

2.1 A Quadratic Programming Problem

In general $f(\mathbf{x})$ is a non-linear and non-convex function, so we have that $L(\mathbf{x}+\boldsymbol{\eta})$ is non-convex in $\mathbf{x}+\boldsymbol{\eta}$, despite being convex in $\mathbf{y}-f(\mathbf{x}+\boldsymbol{\eta})$. Therefore, we propose to relax (1) in order to obtain a convex formulation of the adversarial problem. To that end, let us approximate $f(\mathbf{x})$ by its first order Taylor expansion around $\mathbf{x}$ as $f(\mathbf{x}+\boldsymbol{\eta}) \approx f(\mathbf{x}) + \mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}$. Note that this is a good approximation if $\|\boldsymbol{\eta}\|_p$ is small enough, which is controlled by the $\|\boldsymbol{\eta}\|_p \leq \varepsilon$ constraint. This approximation leads to the following convex approximation of L

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

Since the first three terms of this expression do not depend on $\boldsymbol{\eta}$, the optimization problem from (1) is reduced to

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

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

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

Although this problem is a convex quadratic bowl maximization under an ℓ_p ball constraint and in general challenging, the problem can be solved efficiently in some cases. For the general p, the maximum value is indeed related to the operator norm of $\mathbf{J}_f(\mathbf{x})$. The operator norm of a matrix $\mathbf{A} \in \mathbb{C}^{m\times n}$ between ℓ_p and ℓ_q is defined in Foucart and Rauhut (2013) 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 $\|\mathbf{J}_f(\mathbf{x})\cdot\boldsymbol{\eta}\|_2 = \varepsilon\|\mathbf{J}_f(\mathbf{x})\cdot\frac{\boldsymbol{\eta}}{\varepsilon}\|_2 \leq \varepsilon\|\mathbf{J}_f(\mathbf{x})\|_{p\to 2}$. Therefore, the problem of finding a solution to ($\star$) 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 ℓ_2-norm eigenvector corresponding to the maximum eigenvalue of $\mathbf{J}_f(\mathbf{x})^{\mathrm{T}}\mathbf{J}_f(\mathbf{x})$, then

$$\boldsymbol{\eta}^* = \pm\varepsilon\, \mathbf{v}_{\max} \tag{3}$$

solves the optimization problem. Note that the maximum eigenvalue of $\mathbf{J}_f(\mathbf{x})^{\mathrm{T}}\mathbf{J}_f(\mathbf{x})$ 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 ℓ_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 problem leads to NP-hardness of the problem. Instead the constraint is relaxed by adding ℓ_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

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

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

$$\|\mathbf{J}_f(\mathbf{x})\cdot\boldsymbol{\eta}\|_2 \leq \varepsilon \max_{k\in\{1,\ldots,M\}} \|\mathbf{J}_k\|_2,$$

and the maximum is attained with

$$\boldsymbol{\eta}^* = \pm\varepsilon\mathbf{e}_{k^*} \quad \text{for} \quad k^* = \operatorname*{argmax}_{k\in\{1,\ldots,M\}} \|\mathbf{J}_k\|_2, \tag{4}$$

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 ℓ_∞ 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 Rohn (2000). However, it is possible to approximate this norm using semi-definite programming as proposed in Hartman and Hladík (2015). Semi-definite programming scales badly with input dimension in terms of computational complexity 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 3, where we obtain fast approximate solutions for $\|\mathbf{J}_f(\mathbf{x})\|_{\infty\to 2}$ and single pixel attacks.

2.2 A Linear Programming Problem

The methods derived in Section 2.1 suffer from one main drawback, they require storing $\mathbf{J}_f(\mathbf{x}) \in \mathbb{R}^{K\times N}$ 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 680×480, thus $M = K = 680 \cdot 480 \cdot 3 \approx 9 \cdot 10^5$. 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 $(\star)$ for $p = 2$, we would require computing the eigenvalue decomposition of $\mathbf{J}_f(\mathbf{x})^T\mathbf{J}_f(\mathbf{x})$ as well. This motivates us to relax the problem into a linear programming problem in a similar manner as DeepFool and FGSM, where $\mathbf{J}_f(\mathbf{x})$ is computed implicitly and we do not require to store it. To that end, we relax (1) by directly applying a first order approximation of L, that is $L(\mathbf{x}+\boldsymbol{\eta}) \approx L(\mathbf{x}) + \nabla L(\mathbf{x})^T\boldsymbol{\eta}$. Using this approximation the problem from (1) is now simplified to

$$\max_{\boldsymbol{\eta}} \nabla L(\mathbf{x})^T\boldsymbol{\eta} \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_p \leq \varepsilon\,. \tag{$\bullet$}$$

Note that, for $\mathbf{y} = f(\mathbf{x})$ we have that $\nabla L(\mathbf{x}) = -2\,(\mathbf{y} - f(\mathbf{x}))\,J_f(\mathbf{x})\boldsymbol{\eta}$ is exactly zero, thus leaving this approximation useless for obtaining adversarial perturbations. For this reason, if $\nabla L(\mathbf{x}) = 0$ we take our approximation around another point $\tilde{\mathbf{x}}$ such that $\|\mathbf{x} - \tilde{\mathbf{x}}\|_p \ll \varepsilon$ and $\nabla L(\tilde{\mathbf{x}}) \neq 0$. In other words we approximate L by a linear function around the point $\tilde{\mathbf{x}}$, which is close but not equal to $\mathbf{x}$. This new point $\tilde{\mathbf{x}}$ can be computed by adding a random perturbation, with ℓ_p norm much smaller than ε, to the original point $\mathbf{x}$. Note that changing the center of the first order approximation from $\mathbf{x}$ to $\tilde{\mathbf{x}}$ does not change the nature of the problem since $L(\mathbf{x}+\boldsymbol{\eta}) \approx L(\tilde{\mathbf{x}}) + \nabla L(\tilde{\mathbf{x}})^T(\boldsymbol{\eta} + (\mathbf{x} - \tilde{\mathbf{x}}))$ leads to

$$\max_{\boldsymbol{\eta}} L(\tilde{\mathbf{x}}) + \nabla L(\tilde{\mathbf{x}})^T(\boldsymbol{\eta} - (\mathbf{x} - \tilde{\mathbf{x}})) \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_p \leq \varepsilon \quad \Leftrightarrow \quad \max_{\boldsymbol{\eta}} \nabla L(\tilde{\mathbf{x}})^T\boldsymbol{\eta} \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_p \leq \varepsilon\,.$$

Therefore, for the sake of notation, we assume that $\nabla L(\mathbf{x}) \neq 0$ and refrain from distinguishing between $\mathbf{x}$ and $\tilde{\mathbf{x}}$ for the reminder of this paper. Finally, in the following proposition the closed form solutions to problem $(\bullet)$ are stated which results directly from Hölder's inequality.

Proposition 1. *If $\nabla L(\mathbf{x}) = \left(\frac{\partial L(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial L(\mathbf{x})}{\partial x_M}\right)$, the closed form solution to the problem $(\bullet)$ is given by*

$$\boldsymbol{\eta} = \varepsilon \frac{1}{\|\nabla L(\mathbf{x})\|_q^{q-1}} \times \left(\operatorname{sign}\left(\frac{\partial L(\mathbf{x})}{\partial x_1}\right)\left|\frac{\partial L(\mathbf{x})}{\partial x_1}\right|^{q-1}, \ldots, \operatorname{sign}\left(\frac{\partial L(\mathbf{x})}{\partial x_M}\right)\left|\frac{\partial L(\mathbf{x})}{\partial x_M}\right|^{q-1}\right), \tag{5}$$

where $q = \frac{p}{p-1}$. Particularly, for $p = 1$ the solution is given by the sparse vector

$$\boldsymbol{\eta}^* = \varepsilon \mathbf{e}_{k^*} \quad \textit{with} \quad k^* = \operatorname*{argmax}_k (|(\nabla L(\mathbf{x}))_k|)\,. \tag{6}$$

Corollary 1. *In classification it is common practice to define a classifier $h : \mathbb{R}^M \to \{1, \ldots, K\}$, that assigns the class $h(\mathbf{x}) \in \{1, \ldots, K\}$ to $\mathbf{x}$ as $h(\mathbf{x}) = \operatorname{argmax}_{l\in\{1,\ldots,K\}} \{f_l(\mathbf{x})\}$. Therefore, if we use $L(\mathbf{x}+\boldsymbol{\eta}) = -\min_{l\neq h(\mathbf{x})}\{f_{h(\mathbf{x})}(\mathbf{x}+\boldsymbol{\eta}) - f_l(\mathbf{x}+\boldsymbol{\eta})\}$ instead of the squared loss, the obtained results are also valid for performing adversarial attacks on classification problems. Note that, the adversarial attack manages to change the label of the classifier if and only if $L(\mathbf{x}+\boldsymbol{\eta}) > 0$.*

While these results may be valid for classification problems as well, investigating the performance of this framework for classification is out of the scope of this paper. Therefore, we leave this study for future works.

3 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 (4) and (6) constitute single pixels attacks. This is not true for RGB images where each pixel is represented by a subset of three values. Since we our analysis is not limited to image based systems, we denote these type of attacks as single subset attacks. We need to generalize the single value attacks from (4) and (6) to derive single subset attacks. For this purpose let us partition $\mathcal{M} = \{1, \dots, M\}$ into S possible subsets $\mathcal{S}_1, \dots, \mathcal{S}_S$ of size $Z = |\mathcal{M}|/S$, where $\mathcal{S}_s = \{i_s^1, \dots, i_s^Z\} \subseteq \mathcal{M}$. Using this notation let us define the 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]. Furthermore, $\|\boldsymbol{\eta}\|_{0,\mathcal{S}}$ counts the number of subsets modified by an attacker. This leads to the following formulation of the single subset attack

$$\max_{\boldsymbol{\eta}} L(\mathbf{x}+\boldsymbol{\eta}) \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_\infty \leq \varepsilon\,,\ \|\boldsymbol{\eta}\|_{0,\mathcal{S}} = 1\,, \quad \text{with} \quad L(\mathbf{x}+\boldsymbol{\eta}) = \|\mathbf{y} - f(\mathbf{x}+\boldsymbol{\eta})\|_2^2\,. \tag{7}$$

Note that the norm $\|.\|_{0,\mathcal{S}}$ imposes group sparsity introduced in Rao et al. (2012) as a general case of the ℓ_0 norm for imposing sparsity.

3.1 Single Subset Attack for the Quadratic Problem

As was done in Section 2.1, 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 (7) to

$$\max_{\boldsymbol{\eta}} \|\mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_\infty \leq \varepsilon\,,\ \|\boldsymbol{\eta}\|_{0,\mathcal{S}} = 1\,. \tag{8}$$

Let us first fix the set to be modified by the attacker to s. Then, given s let us define $\boldsymbol{\eta}_s$ as

$$\boldsymbol{\eta}_s = \operatorname*{argmax}_{\boldsymbol{\eta}} \|\mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_\infty \leq \varepsilon\,, (\boldsymbol{\eta})_{i_s^z} = 0\ \forall i_s^z \notin \mathcal{S}_s\,, \tag{9}$$

where $(\boldsymbol{\eta})_{i_s^z}$ denotes the i_s^z-th entry of $\boldsymbol{\eta}$. As discussed in Section 2.1, we know that this type of problem is NP-hard, thus we will search for an approximate solution. Since the maximization of a quadratic bowl over a box constraint lies in one of the corners of the feasible set, we have that $\boldsymbol{\eta}_s = \varepsilon \sum_{z=1}^Z \rho_{i_s^z}^* \mathbf{e}_{i_s^z}$ with $\boldsymbol{\rho}_s^* \triangleq (\rho_{i_s^1}^*, \dots, \rho_{i_s^Z}^*)^T \in \{-1,+1\}^Z$. Therefore, the solution to (8) is

$$\boldsymbol{\eta}^* = \boldsymbol{\eta}_{s*}\,, \quad \text{with} \quad s^* = \operatorname*{argmax}_s \|\mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}_s\|_2^2 \quad \text{and} \quad \boldsymbol{\eta}_s = \varepsilon \sum_{z=1}^Z \rho_{i_s^z}^* \mathbf{e}_{i_s^z} \tag{10}$$

where $\boldsymbol{\rho}_s \triangleq (\rho_{i_s^1}, \dots, \rho_{i_s^Z})^T \in \{-1,+1\}^Z$ and

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

In other words, solving (8) amounts to finding $\boldsymbol{\rho}_s^*$. This problem has the same form as the well known MaxCut problem introduced by Goemans and Williamson (1995). As shown in that work, and subsequent publications, this problem can be relaxed into a semi-definite programming problem. Obtaining an approximate solution of this problem using 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 which is fast but far from optimal. To that end, and without loss of generality, let us assume that for a given s the indices $i_s^1, \dots, i_s^Z$ are sorted such that $\|\mathbf{J}_{i_s^1}\|_2 \geq \dots \geq \|\mathbf{J}_{i_s^Z}\|_2$. We propose an approximate solution for $\rho_{i_s^z}^*$ that is calculated in a greedy manner by setting $\rho_{i_s^1}^* = 1$ and recursively calculating

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

Remark 1. *For the case where $\mathcal{S} = \{\mathcal{M}\}$, the obtained expressions are an approximate solution for ($\star$) under the ℓ_∞ norm constraint on the noise (i.e., $p = \infty$).*

[2] Similar to the so-called ℓ_0-norm, this is not a proper norm.

Type of Attack	Relaxed Problem Formulation	Closed-Form Solution
ℓ_2 / ℓ_∞ constrained	$\max_{\|\boldsymbol{\eta}\|_p=\varepsilon} \|\mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}\|_2^2$	(3) / Remark 1$^\diamond$
	$\max_{\|\boldsymbol{\eta}\|_p=\varepsilon} \boldsymbol{\eta}^T\nabla L(\mathbf{x})$	(5)$^\blacktriangledown$
Single-Subset attack	$\max_{\|\boldsymbol{\eta}\|_\infty=\varepsilon, \|\boldsymbol{\eta}\|_{0,\mathcal{S}}=1} \|\mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}\|_2^2$	(10)$^\diamond$
	$\max_{\|\boldsymbol{\eta}\|_\infty=\varepsilon, \|\boldsymbol{\eta}\|_{0,\mathcal{S}}=1} \boldsymbol{\eta}^T\nabla L(\mathbf{x})$	(12)$^\blacktriangledown$

Table 1: Summary of the obtained closed-form solutions. **Remarks**: ($\blacktriangledown$) valid for regression and classification, ($\diamond$) only an approximate solution for the relaxed problem.

3.2 Single Subset Attack for the Linear Problem

Following the steps from Section 2.2, we make use of the approximation $L(\mathbf{x}+\boldsymbol{\eta}) \approx L(\mathbf{x}) + \nabla L(\mathbf{x})^T\boldsymbol{\eta}$ which leads to the formulation of (7) as a linear programming problem

$$\max_{\boldsymbol{\eta}} \nabla L(\mathbf{x})^T\boldsymbol{\eta} \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_\infty \leq \varepsilon\,,\ \|\boldsymbol{\eta}\|_{0,\mathcal{S}} = 1\,. \tag{11}$$

In the same manner as Section 3.1, for a given s we define $\boldsymbol{\eta}_s$ as in (9). For this linear problem that results in

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

In contrast to the definition of $\boldsymbol{\eta}_s$ from (9), in this case we have a closed form solution for $\boldsymbol{\eta}_s$ as

$$\boldsymbol{\eta}_s = \varepsilon \sum_{z=1}^{Z} \operatorname{sign}((\nabla L(\mathbf{x}))_{i_s^z})\mathbf{e}_{i_s^z}\,,\ \text{thus}\ \nabla L(\mathbf{x})^T\boldsymbol{\eta}_s = \sum_{z=1}^{Z} \left|(\nabla L(\mathbf{x}))_{i_s^z}\right|\,.$$

Therefore, the linear problem for the single subset attack (11) has the closed form solution

$$\boldsymbol{\eta}^* = \boldsymbol{\eta}_{s*}\,,\ \text{with}\ s^* = \operatorname*{argmax}_s \sum_{z=1}^{Z} \left|(\nabla L(\mathbf{x}))_{i_s^z}\right|\ \text{and}\ \boldsymbol{\eta}_s = \varepsilon \sum_{z=1}^{Z} \operatorname{sign}((\nabla L(\mathbf{x}))_{i_s^z})\mathbf{e}_{i_s^z}\,. \tag{12}$$

Remark 2. *The results from* (12) *are valid for classification as well when replacing* L *with* $L(\mathbf{x}+\boldsymbol{\eta}) = -\min_{l\neq k(\mathbf{x})}\{f_{k(\mathbf{x})}(\mathbf{x}+\boldsymbol{\eta}) - f_l(\mathbf{x}+\boldsymbol{\eta})\}$*, in the same manner as Corollary 1.*

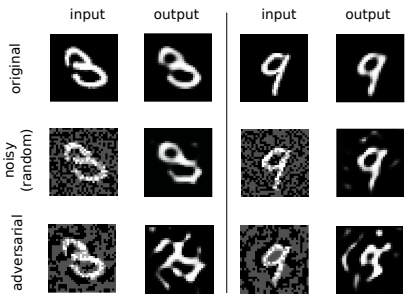
4 Iterative Versions of the Linear Problem

In the previous sections we have formulated several variations of the problem of generating adversarial perturbations. These results are summarized in Table 1. 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 ($\bullet$) is introduced. This corresponds to a gradient ascent method for maximizing $L(\mathbf{x}+\boldsymbol{\eta})$ with a fixed number of iterations and steps of equal ℓ_p norm. While generalizing the results for ($\bullet$) into a gradient ascent method is trivial, the same

Algorithm 1 Iterative extension for ℓ_p constrained methods.

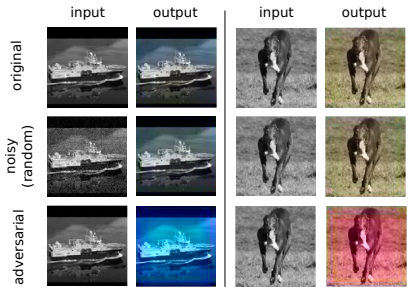
input: $\mathbf{x}$, f, T, ε.
output: $\boldsymbol{\eta}^*$.
Initialize $\mathbf{x}_1 \leftarrow \mathbf{x}$.
for $t = 1, \ldots, T$ **do**
  $\boldsymbol{\eta}_t \leftarrow \operatorname{argmax}_{\boldsymbol{\eta}} \boldsymbol{\eta}^T\nabla L(\mathbf{x}_t)$ s.t. $\|\boldsymbol{\eta}\|_p \leq \varepsilon/T$ (see Table 1)
  $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t + \boldsymbol{\eta}_t$
end for
return: $\boldsymbol{\eta}^* \leftarrow \sum_{t=1}^{T} \boldsymbol{\eta}_t$

is not true for the quadratic problem ($\star$). The main reason for this is that, using the approximation $\mathbf{y} \approx f(\mathbf{x})$, we were able to simplify (2) into ($\star$) since $\mathbf{y} - f(\mathbf{x}) \approx \mathbf{0}$. For an iterative version of this solution we must successively approximate f around different points $\tilde{\mathbf{x}}$, which leads to $\mathbf{y} - f(\tilde{\mathbf{x}}) \neq \mathbf{0}$ even if $\mathbf{y} = f(\mathbf{x})$. We leave the task of investigating alternatives for designing iterative methods with



(a) Autoencoder (96% compression)

(b) Autoencoder (50% compression)



(c) Image Colorization

Figure 1: Adversarial examples for (a): MNIST autoencoder obtained using *quadratic*-ℓ_∞, (b): CIFAR-10 autoencoder obtained using *linear-pixel*-100, (c): STL-10 colorization network obtained using *linear*-ℓ_∞-20.

the results for ($\star$) for future works, and in Section 5 show that the non-iterative solutions for this method are still competitive.

Finally, replacing line 5 of algorithm 1 with

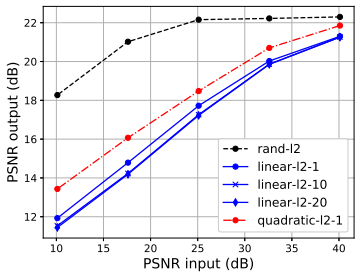
$$\boldsymbol{\eta}_t \leftarrow \underset{\boldsymbol{\eta}}{\operatorname{argmax}}\, \boldsymbol{\eta}^T \nabla L(\mathbf{x}_t) \text{ s.t. } \|\boldsymbol{\eta}\|_p \leq \varepsilon\,, \|\boldsymbol{\eta}\|_{0,\mathcal{S}} = 1$$

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.

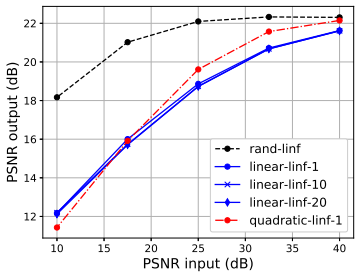
5 Experiments

For the sake of clarity we use the notation *quadratic*-ℓ_p to denote the method of computing adversarial perturbations by solving the quadratic problem ($\star$) under the ℓ_p norm constraint. In the same manner, Algorithm 1 with T iterations and the ℓ_p norm constraint is referred to as *linear*-ℓ_p-T. Since the experiments carried out in this section are exclusively image based, we use the notation *linear-pixel*-T to denote the multiple subset attack with $\|\boldsymbol{\eta}\|_{0,\mathcal{S}} = T$. Since the aim of the proposed attacks is to maximize the MSE of the target system, we use the Peak-Signal-to-Noise Ratio (PSNR), 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.

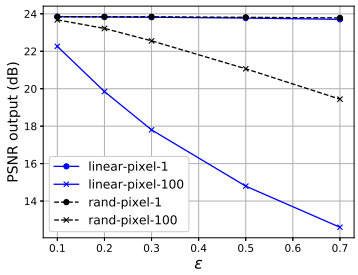
Similarly to Moosavi Dezfooli et al. (2016), we show the validity of our methods by comparing their performance against appropriate types of random noise. For $p = 2$ the random perturbation is computed as $\boldsymbol{\eta} = \varepsilon\, \mathbf{w}/\|\mathbf{w}\|_2$, where the entries of $\mathbf{w}$ are independently drawn from a Gaussian distribution. For $p = \infty$ the random perturbation $\boldsymbol{\eta}$ has independent Bernoulli distributed entries with $\mathbb{P}(\varepsilon) = \mathbb{P}(-\varepsilon) = 1/2$. In the case of multiple subset attacks we perform the same approach as for $p = \infty$ but only on T randomly chosen pixels, while setting the other pixels of $\boldsymbol{\eta}$ to zero. In order to keep a consistent notation, we refer to these 3 methods of generating random perturbations as *random*-ℓ_2, *random*-ℓ_∞, and *random-pixel*-T respectively. For our experiments we use the MNIST, CIFAR-10 and STL-10 datasets. A different neural network is trained for each of these datasets.



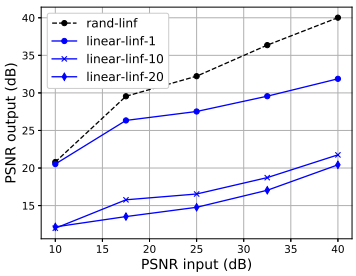
(a) MNIST: ℓ_2 constrained



(b) MNIST: ℓ_∞ constrained



(c) CIFAR-10: Multiple pixel attack



(d) STL-10 (colorization): ℓ_∞ constrained

Figure 2: Output PSNR for (a): MNIST autoencoder under ℓ_2 norm constraint, (b): MNIST autoencoder under ℓ_∞ norm constraint, (c): CIFAR-10 autoencoder under multiple pixel attacks, (d): STL-10 colorization network under ℓ_∞ norm constraint.

As in Tabacof et al. (2016), we also consider autoencoders. For MNIST and CIFAR-10 we have trained fully connected autoencoders with 96% and 50% compression rates respectively. In addition, we go beyond autoencoders and train the image colorization architecture from Baldassarre et al. (2017) for the STL-10 dataset. Different example images obtained from applying the proposed methods on these networks are shown in Figure 1. For instance, in Figure 1(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*-ℓ_∞ method. Similarly, in Figure 1(b), the *random-pixel*-100 algorithm distorts the output significantly more than its random counterpart. These two experiments align with the observation of Tabacof et al. (2016) 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 1(c), where the original and adversarial images are nearly identical.

While the results shown in Figure 1 are for some particular images, in Figure 2 we measure the performance of different adversarial attacks using the average output PSNR over 20 randomly selected images from the corresponding datasets. In Figures 2(a) and 2(b) we observe how computing adversarial perturbations through successive linearizations improves the performance. This behavior is more pronounced in Figure 2(d), where iterative linearizations are responsible for more than 10 dB of output PSNR reduction. Note that, in Figures 2(a) and 2(b) the non-iterative *quadratic*-ℓ_p algorithm performs competitively, even when compared to iterative methods. In Figure 2(b) 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). Finally, in Figure 2(d), we can clearly observe the instability of the image colorization network to adversarial attacks. 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.

References

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

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

Cisse, M., Adi, Y., Neverova, N., and Keshet, J. (2017). Houdini: Fooling deep structured prediction models. *arXiv preprint arXiv:1707.05373*.

Foucart, S. and Rauhut, H. (2013). *A Mathematical Introduction to Compressive Sensing*. Applied and Numerical Harmonic Analysis. Springer New York, New York, NY.

Goemans, M. X. and Williamson, D. P. (1995). Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Hartman, D. and Hladík, M. (2015). Tight Bounds on the Radius of Nonsingularity. In *Scientific Computing, Computer Arithmetic, and Validated Numerics*, Lecture Notes in Computer Science, pages 109–115. Springer, Cham.

Kos, J., Fischer, I., and Song, D. (2017). Adversarial examples for generative models. *arXiv preprint arXiv:1702.06832*.

Lin, Y.-C., Hong, Z.-W., Liao, Y.-H., Shih, M.-L., Liu, M.-Y., and Sun, M. (2017). Tactics of adversarial attack on deep reinforcement learning agents. *arXiv preprint arXiv:1703.06748*.

Metzen, J. H., Kumar, M. C., Brox, T., and Fischer, V. (2017). Universal adversarial perturbations against semantic image segmentation. *stat*, 1050:19.

Miyato, T., Maeda, S.-i., Koyama, M., and Ishii, S. (2017). Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *arXiv preprint arXiv:1704.03976*.

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., and Frossard, P. (2017). Universal adversarial perturbations. *arXiv preprint*.

Moosavi Dezfooli, S. M., Fawzi, A., and Frossard, P. (2016). Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, number EPFL-CONF-218057.

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

Papernot, N., McDaniel, P., Swami, A., and Harang, R. (2016b). Crafting adversarial input sequences for recurrent neural networks. In *Military Communications Conference, MILCOM 2016-2016 IEEE*, pages 49–54. IEEE.

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

Rao, N., Recht, B., and Nowak, R. (2012). Universal Measurement Bounds for Structured Sparse Signal Recovery. In *Artificial Intelligence and Statistics*, pages 942–950.

Rohn, J. (2000). Computing the norm $\|A\|_{\infty,1}$ is NP-hard. *Linear and Multilinear Algebra*, 47(3):195–204.

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

Su, J., Vargas, D. V., and Kouichi, S. (2017). One pixel attack for fooling deep neural networks. *arXiv preprint arXiv:1710.08864*.

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

Tabacof, P., Tavares, J., and Valle, E. (2016). Adversarial images for variational autoencoders. *arXiv preprint arXiv:1612.00155*.

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