Sparse and Imperceptible Adversarial Attack via a Homotopy Algorithm

Mingkang Zhu [1] **Tianlong Chen** [1] **Zhangyang Wang** [1]

Abstract

Sparse adversarial attacks can fool deep neural networks (DNNs) by only perturbing a few pixels (regularized by ℓ_0 norm). Recent efforts combine it with another ℓ_∞ imperceptible on the perturbation magnitudes. The resultant sparse and imperceptible attacks are practically relevant, and indicate an even higher vulnerability of DNNs that we usually imagined. However, such attacks are more challenging to generate due to the optimization difficulty by coupling the ℓ_0 regularizer and box constraints with a non-convex objective. In this paper, we address this challenge by proposing a homotopy algorithm, to jointly tackle the sparsity and the perturbation bound in one unified framework. Each iteration, the main step of our algorithm is to optimize an ℓ_0-regularized adversarial loss, by leveraging the nonmonotone Accelerated Proximal Gradient Method (nmAPG) for nonconvex programming; it is followed by an ℓ_0 change control step, and an optional post-attack step designed to escape bad local minima. We also extend the algorithm to handling the structural sparsity regularizer. We extensively examine the effectiveness of our proposed **homotopy attack** for both targeted and non-targeted attack scenarios, on CIFAR-10 and ImageNet datasets. Compared to state-of-the-art methods, our homotopy attack leads to significantly fewer perturbations, e.g., reducing 42.91% on CIFAR-10 and 75.03% on ImageNet (average case, targeted attack), at similar maximal perturbation magnitudes, when still achieving 100% attack success rates. Our codes are available at: https://github.com/VITA-Group/SparseADV_Homotopy.

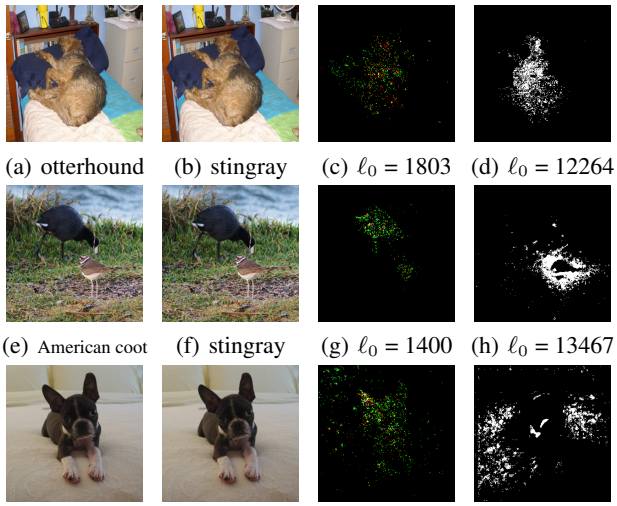

Figure 1. Visualization of pixel-wise sparsity of targeted attack when enforcing a ℓ_∞ constraint of 0.05. The images in each row, from the left to right are benign image, our adversarial example, our perturbation position, GreedyFool (a recent state-of-the-art approach in Dong et al. (2020))'s perturbation position. In the figures of perturbation positions, black pixels denote no perturbation; white pixels denote perturbing all three RGB channels; pure red, green, blue pixels represent perturbing single channel; and pixels with other color denote perturbing two channels.

1. Introduction

Deep neural networks (DNNs) have widely demonstrated fragility to adversarial attacks, e.g., maliciously crafted small perturbations to their inputs that can fool them to make incorrect and implausible predictions (Carlini & Wagner, 2017; Xiao et al., 2018; Athalye et al., 2018; Zhang et al., 2020b; Zhou et al., 2020; Gao et al., 2020; Zhang et al., 2020a; Du et al., 2021). Adversarial attacks raise serious concern against DNNs' applicability in high-stake, risk-sensitive applications. Meanwhile, probing and leveraging adversarial attacks could help us troubleshoot the weakness of a DNN, and further leading to strengthening it, e.g., by adversarial training (Madry et al., 2018).

A non-trivial adversarial attack is meant to be small and "imperceptible" in certain sense (Nguyen et al., 2015): in this way, it impacts little on the benign input's semantic meaning, hence uncovering the inherent instability of the DNN.

[1]The University of Texas at Austin, USA. Correspondence to: Zhangyang Wang <atlaswang@utexas.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

In the most common pixel-level additive attack setting, the standard measure of attack magnitude is by the ℓ_p-norm of perturbations, with $p = 0$, 1, 2, or ∞. In particular, a number of works (Modas et al., 2019; Fan et al., 2020; Xu et al., 2019; Dong et al., 2020; Croce & Hein, 2019) demonstrated that an attack can succeed by only perturbing a few pixels under the ℓ_0 constraint (sometimes ℓ_1). Fig. 1 shows examples of successful pixel-wise sparse targeted attack by our algorithm, compared with a recent state-of-the-art method GreedyFool (Dong et al., 2020). Further, since the vanilla ℓ_0-attacks leave it unconstrained how they change each pixel, the perturbed pixels may have very different intensity or color than the surrounding ones and hence be easily visible (Su et al., 2019). Therefore, more recent methods focus on enforcing some element-wise magnitude constraints to also ensure imperceptible perturbations. In total, such sparse and imperceptible adversarial attacks become a practically relevant topic of interest, indicating an even higher vulnerability of DNNs that we usually imagined, and providing additional insights about interpreting DNN failures (Xu et al., 2019).

1.1. Related Work

Many works on generating sparse and imperceptible adversarial attacks have been proposed recently. The authors of (Croce & Hein, 2019) proposed PGD_0 to project the PGD adversarial attack (Madry et al., 2018) to the ℓ_0 ball, as well as a heuristic CornerSearch strategy by traversing all the pixels and to select a subset to perturb. SparseFool (Modas et al., 2019) relaxes and approximates the ℓ_0 objective using ℓ_1 objective. SAPF (Fan et al., 2020) reformulates the sparse adversarial attack problem into a mixed integer programming (MIP) problem, to jointly optimize the binary selection factors and continuous perturbation magnitudes of all pixels, with a cardinality constraint to explicitly control the degree of sparsity. It was solved by the ℓ_p-Box ADMM designed for integer programming. StrAttack (Xu et al., 2019) focuses on group-wise sparsity and also solves the problem using an ADMM based algorithm. The latest algorithm, GreedyFool (Dong et al., 2020) presents a two-stage greedy solution that first picks perturbations with large gradients and then reduces useless perturbations in the second stage. Besides those white-box attacks, similar problems were also studied in the black-box setting, e.g., One Pixel (Su et al., 2019) and Pointwise Attack (Schott et al., 2019).

Despite progress, challenges remain for those existing solutions. PGD_0 and CornerSearch may fail to yield full success attack rates (Croce & Hein, 2019); they moreover need pre-specified numbers of perturbed pixels, which can also cause inflexible and unnecessarily redundant perturbations. SparseFool (Modas et al., 2019) did not directly control the number of perturbed pixels, and was designed for the non-targeted setting only. Among the latter optimization-based solutions, StrAttack (Xu et al., 2019) and SAPF (Fan et al., 2020) suffer from slow and sometimes unstable convergence in practice, especially when the enforced sparsity is high; it is also not uncommon to observe GreedyFool (Dong et al., 2020) be stuck in local minima because of its greedy nature.

Looking for a more principled optimization solution, we look back into the classical sparse optimization field intensively studied for decades. Among numerous methods proposed, nonconvex regularization based methods, including ℓ_0 and ℓ_p-norm ($0 < p < 1$)-based, have achieved considerable performance improvement over the more traditional convex relaxation (Wen et al., 2018). Specially, iterative hard thresholding methods or proximal gradient methods for ℓ_0-regularized problems are shown to be scalable for (group) sparse regression (Jain et al., 2016). A thorough review of this field can be found in (Bach et al., 2011).

1.2. Our Contributions

This work establishes an ℓ_0-regularized and ℓ_∞-constrained optimization framework, for finding sparse and imperceptible adversarial attacks. Our **technical foundations** are based on an innovative integration between the proximal gradient method for nonconvex optimization described in (Li & Lin, 2015), and the homotopy algorithm. In sparse convex optimization (Soussen et al., 2015; Xiao & Zhang, 2013), the homotopy algorithm (a.k.a. continuation) keeps a sequence of decreasing weights for regularizers, yielding a multistage homotopy solution. In each stage with a fixed weight, the l_0 or l_1-regularized convex optimization problem is optimized. That provides a sparse initial solution for the next stage optimization with a smaller weight, progressively leading to the final sparse solution.

The **core merit** of our method is to enable a smooth relaxation of the sparsity constraint, which eventually leads to a flexible yet compactly sparse solution. We outline our specific contributions below:

- We first introduce the nonmonotone Accelerated Proximal Gradient Method (nmAPG) to jointly tackle the sparsity and the box constraint. The algorithm can also be extended to handling *group-wise sparsity*. A main hurdle in this regularization-based method is yet how to achieve the desired sparsity: this is typically controlled by some ad-hoc chosen regularization coefficient.
- We then integrate the homotopy algorithm into our framework, which allows us to optimize using a sequence of gradually decreasing weights for regularizers (Soussen et al., 2015; Xiao & Zhang, 2013). It can continuously provide a sparse initial solution for the next stage optimization, which leads to a compact sparse solution by the end. To deal with the weight sensitivity of the ℓ_0-regularizer in the homotopy algorithm, we design an ℓ_0-change control step on the perturbation update at each iteration, and another optional step called

post-attack perturbation to escape bad local minima. Both lead to stable sparse solutions without extensive weight searching for regularization-based methods.

- Extensive experiments on the CIFAR-10 (Krizhevsky, 2009) and ImageNet (Deng et al., 2009) endorse the superiority of our new **homotopy attack**. In both targeted and non-targeted attack scenarios, it outperforms state-of-the-art methods by significantly fewer perturbations, e.g., reducing 42.91% on CIFAR-10 and 75.03% on ImageNet (average case, targeted attack), at similar perturbation magnitudes, when still achieving 100% attack success rates. Visualizing the attacks also gains us more interpretation of DNN fragility.

2. Method

2.1. Problem Formulation

Let $\mathcal{X}$ be the set of benign images. A trained DNN image classification model maps each image $x \in \mathbb{R}^n$ from $\mathcal{X}$ to its class label $y \in \mathcal{Y} = \{1, 2, \cdots, K\}$. Given a benign image $x_0 \in \mathcal{X}$, a *targeted* sparse and imperceptible adversarial attack aims for finding a perturbation $\delta \in \mathbb{R}^n$, so that the perturbed image $x_0 + \delta$ is incorrectly classified to a target class t. In the meanwhile, the number of perturbations should be as sparse as possible, and the perturbation magnitude should remain small. This problem can be cast as

$$\begin{aligned} \min_{\delta} \quad & f(x_0+\delta, t) + \lambda ||\delta||_0 \\ s.t. \quad & ||\delta||_\infty \leq \epsilon,\ 0 \leq x_0 + \delta \leq 1. \end{aligned} \tag{1}$$

Here $f()$ is the classification loss such as the cross entropy function, λ is the regularization coefficient to control the ℓ_0 sparsity of δ, and ϵ is the maximal allowable perturbation magnitude. This above targeted attack formulation can be easily adapted to the *nontargeted* attack by replacing the adversarial loss $f()$:

$$\begin{aligned} \min_{\delta} \quad & f(x_0+\delta, y_0) + \lambda ||\delta||_0 \\ s.t. \quad & ||\delta||_\infty \leq \epsilon,\ 0 \leq x_0 + \delta \leq 1. \end{aligned} \tag{2}$$

Since the algorithms for solving (1) and (2) will have most in common, we will focus our discussion on (1) hereinafter, while noting that all algorithmic steps can be readily utilized towards solving (2) unless otherwise specified.

2.2. A Basic Solution using Accelerated Proximal Gradient for Nonconvex Optimization

Since both constraints in (1) are box constraints, they can be simplified to one box constraint as $\delta \in [l, u]$, where $l, u \in \mathbb{R}^n$, and $0 \in [l, u]$. Let $I_{[l,u]}$ be the indicator function such that $I_{[l,u]}(\delta) = 0$ if $\delta \in [l, u]$, and $I_{[l,u]}(\delta) = +\infty$ otherwise. Problem (1) can be rewritten as

$$\min_{\delta} f(x_0+\delta, t) + \lambda ||\delta||_0 + I_{[l,u]}(\delta). \tag{3}$$

The non-convexity and non-smoothness make solving (3) a nontrivial task. The classical accelerated proximal gradient (APG) methods (Nesterov, 2004) have been proven to be efficient for convex optimization problems, but it remains unclear whether the usual APG can converge to a critical point of nonconvex programming. In (Li & Lin, 2015), the authors proposed two convergent accelerated proximal gradient methods (APG) for nonconvex and nonsmooth programs, of which (3) is a special case. In this paper, we adopt their proposed *nonmonotone APG with line search* (nmAPG) algorithm.

To be self-containing, we briefly describe the high-level idea, and refer the readers to (Li & Lin, 2015) for more details. APG first obtains an extrapolation solution using the current and the previous solutions, and then solve proximal mapping problems. When extending APG to nonconvex programming, the main challenge is to prevent bad extrapolation due to oscillatory landscapes (Beck & Teboulle, 2009). The authors of (Li & Lin, 2015) introduced a monitor to prevent and correct bad extrapolations so that they satisfy the sufficient descent property. In practice, nmAPG searches for a step size using the backtracking line search initialized with the BB step-length (Barzilai & Borwein, 1988) since the Lipstchiz constant of gradients is not exactly known.

We next lay out how to solve (3) with nmAPG. Suppose that f is a smooth nonconvex function, whose gradient has Lipschitz constant L. Given δ^k, it holds that

$$\begin{aligned} f(x_0+\delta, t) \leq & f(x_0+\delta^k, t) + \\ & \nabla_\delta f(x_0+\delta^k, t)^T(\delta - \delta^k) + \tfrac{L}{2}||\delta - \delta^k||^2, \end{aligned}$$

where $||\cdot||$ is the $||\cdot||_2$ norm for simplicity.

Then we consider the problem

$$\begin{aligned} \min_{\delta} \quad & f(x_0+\delta^k, t) + \nabla_\delta f(x_0+\delta^k, t)^T(\delta - \delta^k) + \\ & \tfrac{L}{2}||\delta - \delta^k||^2 + \lambda||\delta||_0 + I_{[l,u]}(\delta), \end{aligned}$$

which is equivalent to

$$\begin{aligned} \min_{\delta} \quad & \tfrac{L}{2}||\delta - [\delta^k - \tfrac{1}{L}\nabla_\delta f(x_0+\delta^k, t)]||^2 + \\ & \lambda||\delta||_0 + I_{[l,u]}(\delta). \end{aligned} \tag{4}$$

Now letting $g(\delta) = \lambda||\delta||_0 + I_{[l,u]}(\delta)$, we denote the solution to (4) as

$$\begin{aligned} \text{Prox}_{\frac{1}{L}g}(\delta^k - \tfrac{1}{L}\nabla_\delta f(x_0+\delta^k, t)) = \operatorname*{argmin}_{\delta} \frac{L}{2}||\delta - [\delta^k - \\ \tfrac{1}{L}\nabla_\delta f(x_0+\delta^k, t)]||^2 + \lambda||\delta||_0 + I_{[l,u]}(\delta). \end{aligned} \tag{5}$$

In fact, it can be obtained explicitly. Let

$$S_L(\delta) = \delta - \frac{1}{L}\nabla_\delta f(x_0+\delta, t), \forall \delta \in [l, u],$$

and

$$\Pi_{[l,u]}(\delta) = \text{argmin}\{||y - \delta|| : y \in [l, u]\}, \forall \delta \in \mathbb{R}^n.$$

It is easy to obtained that (Lu, 2013), the optimal solution of problem (4) is (for $i = 1, 2, \cdots, n$)

$$\delta_i^{k+1} = \begin{cases} [\Pi_{[l,u]}(s_L(\delta^k))]_i, & \text{if } [s_L(\delta^k)]_i^2 - [\Pi_{[l,u]}(\\ & s_L(\delta^k)) - s_L(\delta^k)]_i^2 > \frac{2\lambda}{L}; \\ 0, & \text{otherwise,} \end{cases}$$

Since $g(\delta) = \lambda||\delta||_0 + I_{[l,u]}(\delta)$ is a proper and lower semi-continuous function, the convergence of nmAPG to a critical point of problem (3) can be assured, under some mild conditions (Li & Lin, 2015).

2.3. Extension to Group-wise Sparsity

Group-wise sparsity has been recently incorporated into adversarial attack by the recent work (Xu et al., 2019) to provide more structured and explainable perturbations. To extend our algorithm from the element-wise ℓ_0 regularizer to the group-wise sparsity, we need to reformulate the problem and adapt the nmAPG solution.

Suppose the input image x_0 can be partitioned into m groups $x^T = (x_{G_1}^T, \cdots, x_{G_m}^T)$. Then the problem concerning group-wise sparsity and imperceptibility can be cast as

$$\begin{array}{ll} \min\limits_{\delta} & f(x_0 + \delta, t) + \lambda||\delta||_{2,0} \\ s.t. & ||\delta||_\infty \le \epsilon,\ 0 \le x_0 + \delta \le 1, \end{array} \tag{6}$$

where $||\delta||_{2,0}$ is defined as

$$||\delta||_{2,0} = |\{i : ||\delta_{G_i}|| \neq 0, i = 1, 2, \cdots, m\}|.$$

By similarly using the indicator function $I_{[l,u]}(\delta)$, problem (6) can be further written equivalently as

$$\min_{\delta} f(x_0 + \delta, t) + \lambda||\delta||_{2,0} + I_{[l,u]}(\delta). \tag{7}$$

Suppose that f is a smooth nonconvex function, whose gradient is Lipschitzian with Lipschitz constant L. Similar to (4), given δ^k, we consider the problem

$$\begin{aligned} &\min_{\delta} \frac{L}{2}||\delta - [\delta^k - \frac{1}{L}\nabla_\delta f(x_0 + \delta^k, t)]||^2 + \lambda||\delta||_{2,0} \\ &\qquad + I_{[l,u]}(\delta) \\ &= \min_{\delta} \frac{L}{2}||\delta - s_L(\delta^k)||^2 + \lambda||\delta||_{2,0} + I_{[l,u]}(\delta). \end{aligned} \tag{8}$$

In (Beck & Hallak, 2019), the authors presented the solution to a problem more general than (8). Although their derivation can be applied, we still exhibit the concise solution to problem (8) and give a simple proof, for completeness.

Proposition 2.1. *The minimum solution of problem* (8) *is*

$$\delta_{G_i}^{k+1} = \begin{cases} [\Pi_{[l,u]}(s_L(\delta^k))]_{G_i}, & \textit{if } ||[s_L(\delta^k)]_{G_i}||^2 - \\ & ||[\Pi_{[l,u]}(s_L(\delta^k)) - s_L(\\ & \delta^k)]_{G_i}||^2 > \frac{2\lambda}{L}; \\ 0, & \textit{otherwise,} \end{cases} \tag{9}$$

for $i = 1, 2, \cdots, m$.

Proof. First, due to group-wise separability, Equation (8) can be decomposed into m subproblems ($i = 1, 2, \cdots, m$):

$$\min_{\delta_{G_i}} \frac{L}{2}||[\delta - s_L(\delta^k)]_{G_i}||^2 + \lambda \cdot \text{sgn}(\|\delta_{G_i}\|) + I_{[l_{G_i}, u_{G_i}]}(\delta_{G_i}), \tag{10}$$

If $||\delta_{G_i}|| = 0$, which means $\delta_{G_i} = 0$, then the objective value of problem (10) is $\frac{L}{2}||[s_L(\delta^k)]_{G_i}||^2$;

If $||\delta_{G_i}|| \neq 0$, then $\text{sgn}(\|\delta_{G_i}\|) = 1$, and problem (10) becomes

$$\lambda + \sum_{j \in G_i} \min_{\delta_j} \frac{L}{2}(\delta_j - [s_L(\delta^k)]_j)^2 + I_{[l_j, u_j]}(\delta_j). \tag{11}$$

In (11), it is obvious that the minimum solution is $[\Pi_{[l,u]}(s_L(\delta^k))]_j, \quad \forall j \in G_i$, which can be unified as $[\Pi_{[l,u]}(s_L(\delta^k))]_{G_i}$. And the minimum value of (11) is

$$\lambda + \sum_{j \in G_i} \frac{L}{2}([\Pi_{[l,u]}(s_L(\delta^k)) - s_L(\delta^k)]_j)^2,$$

which can be written simply as $\lambda + \frac{L}{2}||[\Pi_{[l,u]}(s_L(\delta^k)) - s_L(\delta^k)]_{G_i}||^2$. Thus by comparing the two objectives, the solution of problem (8) can be obtained as (9). □

Now, let $g(\delta) = \lambda||\delta||_{2,0} + I_{[l,u]}(\delta)$. We denote the solution of problem (8) as

$$\begin{aligned} &\text{Prox}_{\frac{1}{L}g}(\delta^k - \tfrac{1}{L}\nabla_\delta f(x_0 + \delta^k, t)) \\ &= \text{argmin}_{\delta} \frac{L}{2}||\delta - [\delta^k - \frac{1}{L}\nabla_\delta f(x_0 + \delta^k, t)]||^2 + \lambda||\delta||_{2,0} \\ &\qquad + I_{[l,u]}(\delta). \end{aligned}$$

Note that $g(\delta) = \lambda||\delta||_{2,0} + I_{[l,u]}(\delta)$ is also a proper and lower semicontinuous function, the convergence of nmAPG to a critical point can similarly be assured just like in the element-wise sparsity case.

2.4. Putting Homotopy into Our Solution

Existing sparse adversarial attack methods that use regularized optimization generally pre-select an ad-hoc fixed weight for the regularization term that controls sparsity (Fan et al., 2020; Xu et al., 2019). This setting is over-idealistic, and in practice requires multiple trial-and-errors on tuning the weight in order to achieve certain desired sparsity level. To alleviate this issue, we format our nmAPG based algorithm in a **homotopy** manner (Xiao & Zhang, 2013; Lin & Xiao, 2014), which allows us to optimize using a sequence of decreasing weights for regularizers. It can keep a multi-stage homotopy solution; in each stage, we optimize (1) using nmAPG until it reaches the maximum inner iterations.

That can continuously provide a sparse initial solution from the last stage for the next stage optimization, leading to the final sparse solution.

Despite homotopy being a highly effective and flexible strategy for convex optimization, it might not be directly applied to highly nonconvex problems with high dimension such as the adversarial attack of a DNN. For example, even if an input perturbation into the nmAPG for problem (1) is sparse, the converged solution by nmAPG might be a bad local minimum and might not be sparse, since the optimized function is highly nonconvex. Below, we describe several strategies that we propose to stabilize and improve the homotopy algorithm for our problem.

Initial Weight Search: The initial value of the weight λ is important for subsequent performance. For homotopy, an ideal initial weight in problem (1) should produce a solution δ near 0 in the initial iteration of nmAPG. This can be simply determined by increasing its value in a coarse-grained manner until the nmAPG with only the first iteration gives $\delta = 0$, since if λ is large enough, nmAPG will focus on optimizing the sparsity function. Then the value is decreased in a relatively fine-grained manner until the first iteration of nmAPG starts to update on δ. The obtained value is finally multiplied by a constant c and is assigned as the initial weight λ_0, to better enforce sparsity constraint. Algorithm 1 summarizes our strategy in details.

Algorithm 1 Our Subroutine for Initial Weight Search (Lambda_Search)

input x_0: benign image; t: target;
eta, $delta$, rho: parameters of nmAPG;
c, v, β: parameters of this algorithm.
output λ.

$\delta^0 = 0$; $\lambda = \beta$;
repeat
  δ^1=nmAPG(δ^0, x_0, t, λ, eta, $delta$, rho, v, MaxIter=1);
  $\lambda = \lambda + \beta$;
until $\delta^1 = 0$;
repeat
  δ^1=nmAPG(δ^0, x_0, t, λ, eta, $delta$, rho, v, MaxIter=1);
  Decrease λ by a factor;
until $\delta^1 \neq 0$;
$\lambda = \lambda * c$.

Additional Sparsity Control by ℓ_0 Norm Changes: Even with a good initial weight, the homotopy solution with ℓ_0 regularizer remains to be sensitive to the regularization parameter λ along the homotopy path, which may not suffice for precise sparsity control for problem (1) when both the loss function and the regularizer are nonconvex. Moreover, due to the same reason, for the problem (1) with a fixed weight λ, even the input solution of nmAPG is sparse, the converged solution might not be sparse. We hereby introduce an additional control for sparsity.

Specifically, we constrain the maximum number of ℓ_0 changes to be v in every outer iteration of homotopy. Suppose r is the ℓ_0-norm of the input perturbation δ to nmAPG in the current outer iteration of homotopy. Then, at the end of every inner iteration of nmAPG, we simply keep the entries of δ with the top $(r + v)$ absolute values and reduce all other entries to 0. This will keep the number of nonzero entries of the perturbation increased steadily, which avoids the situation that a small decrease of the weight λ makes the ℓ_0-norm of the perturbation increase suddenly.

The value of v is an input constant of our homotopy algorithm. However, it will be updated temporarily as a small positive integer to be input to nmAPG, whenever the current perturbation δ^{k+1} satisfies that

$$\frac{||\delta^{k+1}||_1}{||\delta^{k+1}||_0} \leq \epsilon * \gamma, \tag{12}$$

where $\gamma < 1$ is a user-specified constant. An intuition behind this mechanism is that, if inequality (12) is satisfied, then the average perturbation magnitude of δ^{k+1} is much smaller than the invisible threshold ϵ, which means the number of $||\delta^{k+1}||_0$ entries has not been fully used to attack the classification model at present. Hence v is set temporarily to a small positive integer, and nmAPG will solve problem (1) with a smaller v until inequality (12) is violated.

Optional Post-Attack Perturbation: The above scheme can already generate sparse adversarial perturbations. However, we observe that our homotopy algorithm sometimes gets stuck in bad local minima, which is inevitable due to the nonconvex f. In this case, as λ decreases in the homotopy algorithm, the ℓ_1-norm of the perturbation increases disproportionally to the corresponding ℓ_0-norm, which means the algorithm cannot effectively use the perturbed entries, and inequality (12) is satisfied. So inequality (12) is regarded as a trigger condition that our algorithm falls into a local minimum after the nmAPG stage.

To alleviate this issue, we intend to provide some "push" to help our algorithm escape from the local minimum faster. In each outer iteration of the homotopy, we check whether inequality (12) is satisfied. If it is the case, we propose to execute the following post-attack perturbation step:

$$\begin{aligned} \min_{\delta} \quad & w_1 f(x_0 + \delta, t) + w_2 ||\delta||_p \\ s.t. \quad & ||\delta||_\infty \leq \epsilon,\ 0 \leq x_0 + \delta \leq 1, \end{aligned} \tag{13}$$

where $w_1 \gg w_2$, p can be either 1, 2, or ∞, and δ is now enforced only on **nonzero entries** of the output perturbation of the nmAPG stage.

The assumption that $w_1 \gg w_2$ makes an optimizer focus on minimizing the function f, and the value of $||\delta||_p$ might increase. Then the inequality (12) might be violated, which means our algorithm might have escaped from the bad local minimum. Since we only need to provide some push

directions rather than precise updates, we simply optimize problem (13) using gradient descent for a number of iterations proportional to the current ℓ_0. After the post attack stage, combined with the nmAPG stage with smaller v in the next outer iteration of homotopy, the ℓ_1-norm of the perturbation would increase such that inequality (12) is violated, and then we can set v back to normal.

Our full **homotopy attack** algorithm are summarized as Algorithm 2. Starting with the initial weight search, the algorithm has an outer iteration by homotopy. Within each outer iteration, the main step is to optimize an ℓ_0-regularized loss by leveraging nmAPG (which will have its inner iterations), followed by an additional ℓ_0 change control step, and an optional post-attack to escape from bad local minima.

Algorithm 2 The Homotopy Attack Algorithm

input x_0: benign image; t: target; ϵ: invisible threshold;
eta, $delta$, rho, MaxIter: parameters of nmAPG;
c, v, β, γ: parameters of this algorithm.
output δ.

$\delta^0 = 0$; $v_{ini} = v$; $k = 0$;
λ =Lambda_Search(x_0, t, c, eta, $delta$, rho, v, β);
repeat
  δ^{k+1} = nmAPG(δ^k, x_0, t, λ, eta, $delta$, rho, v, MaxIter);
  $v = v_{ini}$;
  if not success **then**
    if $||\delta^{k+1}||_1 \leq ||\delta^{k+1}||_0 * \epsilon * \gamma$ **then**
      Set v as a small integer;
      Conduct post-attack perturbation;
    end if
    Decrease λ by a factor;
  end if
  $k = k + 1$;
until Attack Succeeds.

3. Experiments

In this section, we conduct comprehensive experiments with diverse setups to validate the effectiveness of proposed homotopy algorithm on the CIFAR-10 (Krizhevsky, 2009) and the ImageNet (Deng et al., 2009) datasets. Section 3.1 collects our experimental setups. Then, we compare multiple previous state-of-the-art (SOTA) sparse adversarial attack approaches in Section 3.2, 3.3, and 3.4, including GreedyFool (Dong et al., 2020), SAPF (Fan et al., 2020), and StrAttack (Xu et al., 2019). Moreover, ablation studies, the application to group-wise sparsity, empirical time cost, and visualization are provided in Section 3.5, 3.6, 3.7, and 3.8, respectively.

3.1. Experiment Setup

Dataset and Classification Model Setting: Following the practice of (Xu et al., 2019) and (Fan et al., 2020), we randomly select 1000 images from the test set of CIFAR-10 for targeted attack. Each selected image would be attacked with 9 target classes except its own class label, thus 9000 adversarial examples are generated in total. As for the ImageNet dataset, we select randomly 100 images from the validation set and 9 classes as the targets for targeted attack, so we generate 900 adversarial examples from the ImageNet dataset. For nontargeted attack, we randomly select 5000 images from the test set of CIFAR-10, and 1000 images from the validation set of ImageNet as the input images.

For the classification model on CIFAR-10 dataset, we follow the practice of (Fan et al., 2020), (Carlini & Wagner, 2017), and (Xu et al., 2019) to train a network that consists of four convolution layers, two max-pooling layers, and two fully-connected layers. Our trained network can achieve 80% classification accuracy on the CIFAR-10 dataset. For the classification model on the ImageNet dataset, we choose to use the pretrained Inception-v3 model (Szegedy et al., 2016), which can achieve 77.45% top-1 classification accuracy and 96% top-5 classification accuracy on the ImageNet dataset.

Parameter Setting: For targeted attack, the adversarial loss function $f()$ is set as the cross entropy function. For nontargeted attack, we adopt the loss function proposed in (Carlini & Wagner, 2017) directly:

$$f(x_0, y_0) = \max\{D(x_0)_{y_0} - \max_{i \neq y_0}\{D(x_0)_i\}, -\kappa\},$$

where x_0 is a benign image and y_0 is its ground-truth class label, D is the model to attack, and $D(x_0)_{y_0}$ is the logit for class y_0. The confidence parameter κ is set to 0.

Since we are highly interested in generating sparse and invisible adversarial perturbations while not extremely sparse but visible ones, we maintain a relatively small ℓ_∞-norm of generated perturbations. That is, we set ϵ to 0.05, which is a relatively small number in the $[0, 1]$ range of a valid image.

While, except GreedyFool (Dong et al., 2020), the other two compared methods (Fan et al., 2020; Xu et al., 2019) do not have a control on the ℓ_∞ of generated perturbations. They either use a fixed number as an upper bound on the ℓ_0-norm, or adopt a fixed weight for sparsity regularizer. When enforcing a strict sparsity constraint, their methods may fail or cause the ℓ_∞ of generated perturbations to be extremely large, which is perceptible by human eyes. Therefore, to establish a fair comparison, we use our ϵ setting as a reference, and tune the parameters of their methods to ensure that the average ℓ_∞ of generated perturbations is close to ϵ. For GreedyFool (Dong et al., 2020), we use their original implementation and set the upper bound on ℓ_∞ to 0.05. The similar level of ℓ_∞ enables us to make a fair comparison on the ℓ_0 of generated perturbations.

Other parameters of our algorithm and the parameters for group-wise sparsity are discussed in detail in the appendix.

Evaluation Metrics: We report the average ℓ_p-norms of generated perturbations, where $p = 0, 1, 2, \infty$, and the attack success rates (ASR) of the experimented methods with

Table 1. Statistics of attack success rate and average ℓ_p-norms ($p = 0, 1, 2, \infty$) of targeted attack on CIFAR-10 and ImageNet datasets.

Database	Method	Best case					Average case					Worst case				
		ASR	ℓ_0	ℓ_1	ℓ_2	ℓ_∞	ASR	ℓ_0	ℓ_1	ℓ_2	ℓ_∞	ASR	ℓ_0	ℓ_1	ℓ_2	ℓ_∞
CIFAR-10	GreedyFool	100	199	6.583	0.468	0.048	100	289	10.037	0.606	0.049	100	445	15.935	0.786	0.049
	SAPF	100	314	4.067	0.258	0.051	100	518	7.487	0.362	0.051	100	594	9.860	0.452	0.051
	StrAttack	100	363	4.570	0.248	0.050	100	546	8.267	0.501	0.054	100	657	10.843	0.485	0.056
	Homotopy-Attack	100	110	5.166	0.467	0.047	100	165	7.955	0.580	0.049	100	234	11.310	0.686	0.049
ImageNet	GreedyFool	100	8937	283.705	3.287	0.049	100	10837	290.848	3.176	0.049	100	12197	330.049	3.552	0.049
	SAPF	100	37923	84.726	0.852	0.056	100	54743	117.488	0.980	0.054	100	60374	143.718	1.179	0.062
	StrAttack	100	39593	95.122	1.039	0.061	100	55410	136.314	1.144	0.056	100	61328	165.434	1.203	0.067
	Homotopy-Attack	100	2399	99.178	2.030	0.049	100	2706	111.697	2.134	0.049	100	3065	126.933	2.288	0.050

both the targeted and nontargeted settings. In the targeted setting, we follow Fan et al. (2020) and Xu et al. (2019) to separate the results into three cases: the best case, the average case and the worst case. The best case presents the results of target class that leads to the most sparsity, the worst case shows the results of target class that leads to the least sparsity, and the average case gives the average results of all 9 target classes.

3.2. Targeted Attack

Table 1 presents the results of targeted attack on the CIFAR-10 and ImageNet, where "Homotopy-Attack" denotes our proposed algorithm. From this table, we can see that all compared methods can achieve 100% attack success rate. However, our algorithm outperforms the other compared methods on sparsity level by a large margin. Specifically, our algorithm can generally achieve 5.4% sparsity on $32 \times 32 \times 3$ CIFAR-10 images and 1% sparsity on $299 \times 299 \times 3$ ImageNet images in average case, which is 42.91% fewer on CIFAR-10 and 75.03% fewer on ImageNet, respectively, when compared to SOTA.

From Table 1, a noticeable finding is that, the ratios of ℓ_1 to ℓ_0 of our algorithm are generally larger than the other three methods on both the two datasets. We believe this is not a bad phenomenon. To enforce higher level of sparsity, there needs to be trade-offs between the ℓ_0-norm and the ℓ_1 and ℓ_2-norms. Further, we can discover from Table 1 that, the average ratios of ℓ_1 to ℓ_0 of our generated perturbations are generally higher than 0.04, which is very close to the ℓ_∞-threshold ϵ. This result indicates that our algorithm is able to make the most trade-offs between the ℓ_0-norm and the ℓ_1 and ℓ_2-norms, which can lead to sparser results. This may also indicate that relaxing the ℓ_0 constraint to ℓ_1 may not be optimal in the case of sparse adversarial attack.

3.3. Nontargeted Attack

GreedyFool (Dong et al., 2020) outperforms the state-of-the-art methods like SparseFool (Modas et al., 2019) and PGD_0 (Croce & Hein, 2019) by a significant large margin on nontargeted attack. Since SAPF (Fan et al., 2020) adopted a similar optimization method and achieved better results than StrAttack (Xu et al., 2019) in Table 1, we focus on comparing compare our algorithm with GreedyFool and SAPF on nontargeted attack.

Experimental results of nontargeted attack on the ImageNet and CIFAR-10 datasets are listed in Table 2. From this table, we can see that all methods achieve 100% attack success rate. Looking at the ℓ_p-norms, we can find that our algorithm on average only need to perturb 2.3% entries on the $32 \times 32 \times 3$ images from CIFAR-10 dataset, and 0.14% entries on the $299 \times 299 \times 3$ images from ImageNet dataset, which is 37.93% fewer on CIFAR-10 and 65.69% fewer on ImageNet, respectively, when compared to SOTA.

Further, we can find from Table 2 that, the ratios of ℓ_1 to ℓ_0 of our algorithm are still larger than 0.04, and our algorithm can generate much sparser result. Thus the findings mentioned in Subsection 3.2 also apply for nontargeted attack.

Table 2. Statistics of attack success rate and average ℓ_p-norms ($p = 0, 1, 2, \infty$) of nontargeted attack on CIFAR-10 and ImageNet.

Database	Method	ASR	ℓ_0	ℓ_1	ℓ_2	ℓ_∞
CIFAR-10	GreedyFool	100	116	4.395	0.392	0.048
	SAPF	100	352	6.070	0.365	0.056
	Homotopy-Attack	100	72	3.427	0.367	0.049
ImageNet	GreedyFool	100	1128	35.764	0.964	0.049
	SAPF	100	2063	24.907	0.637	0.058
	Homotopy-Attack	100	387	16.387	0.685	0.047

3.4. Scale Up to Stronger Backbone Network

To further validate our method's effectiveness on CIFAR-10 dataset, we train a ResNet-18 (He et al., 2016) which can achieve 93% accuracy on CIFAR-10 dataset, and conduct the previous experiments with GreedyFool (Dong et al., 2020) and SAPF (Fan et al., 2020). The results are listed in Tables 3. From the table, we can see that our algorithm outperforms GreedyFool (SOTA) by a larger margin. Our algorithm is able to reduce 61.46% sparsity, and 62.87% sparsity respectively on nontargeted and targeted attacks when compared to GreedyFool. It indicates that our algorithm is more effective than greedy-based methods when attacked networks become stronger.

Table 3. Statistics of targeted and nontargeted attacks on ResNet-18 on CIFAR-10.

Attack Type	Method	ASR	ℓ_0	ℓ_1	ℓ_2	ℓ_∞
Nontargeted	GreedyFool	100	192	6.192	0.439	0.048
	SAPF	100	723	7.758	0.345	0.055
	Homotopy-Attack	100	74	3.405	0.365	0.047
Targeted	GreedyFool	100	668	19.667	0.773	0.048
	SAPF	100	1320	11.801	0.437	0.071
	Homotopy-Attack	100	248	11.359	0.676	0.049

3.5. Ablation Study

To further understand the contribution of each component in our algorithm, ablation studies are conducted on the CIFAR-10 and ImageNet datasets. Without loss of generality, we only experiment on targeted attack, since it is more difficult than nontargeted attack. The results are listed in Table 4, in which we term the homotopy algorithm without additional sparsity control and optional post attack stage as "Pure-Homotopy", and the nmAPG algorithm with binary weight search as "nmAPG".

From Table 4, we can see that both nmAPG with binary weight search and homotopy without additional sparsity control or post attack stage perform much worse than our homotopy attack algorithm. The sparsity levels are improved significantly after adding the additional control and post attack stage to Pure-Homotopy on both datasets. We believe there are 2 reasons behind this: (1) The ℓ_0-norm regularizer is nonsmooth and nonconvex, and it is sensitive to the change of weight λ in either the homotopy manner or the binary search manner. So we cannot be certain about if the decrease of λ in each iteration of the homotopy algorithm will cause a significant change of the ℓ_0-norm, which we do not want it to happen. So further control of the maximum increase of ℓ_0 is needed here. (2) Since the loss function f is nonconvex, the optimization can easily be trapped in local minima. If that happens, we need to either push it out or give it some directions. Without the "push", the algorithm would require a huge increase of the ℓ_0-norm to escape. So the post attack stage is needed.

Table 4. Statistics of attack success rate and average ℓ_p-norms ($p = 0, 1, 2, \infty$) of component analysis experiment.

Database	Method	ASR	ℓ_0	ℓ_1	ℓ_2	ℓ_∞
CIFAR-10	Pure-Homotopy	100	223	10.087	0.644	0.050
	nmAPG	100	283	12.687	0.714	0.050
	Homotopy-Attack	100	168	7.409	0.555	0.049
ImageNet	Pure-Homotopy	100	16343	340.691	3.094	0.050
	nmAPG	100	13582	303.195	2.998	0.050
	Homotopy-Attack	100	2889	117.97	2.209	0.049

3.6. Group-wise Sparsity

In this subsection, we show the group-wise sparsity experimental results of our algorithm on targeted attack. The results are given in Table 5. Since we only constrain group-wise sparsity, the average ℓ_0-norm is larger than the corresponding one in Table 1, where we conduct pixel-wise sparsity. However, we can see from the $\ell_{2,0}$ that, the group-wise sparsity level is very high. Specifically, we can on average achieve 6.88% and 2.05% group sparsity on CIFAR-10 and ImageNet, respectively.

Table 5. Statistics of our algorithm on group-wise sparse targeted attack.

Database	ASR	ℓ_0	$\ell_{2,0}$	ℓ_1	ℓ_2	ℓ_∞	no. of groups
CIFAR-10	100	502	5.501	19.407	0.881	0.047	80
ImageNet	100	7711	8.667	292.862	3.461	0.048	423

3.7. Empirical Time Cost

In this subsection, we compare the running time of our algorithm with state-of-the-art methods. The results are given in Table 6. From the table we can see that our algorithm is much faster than SAPF (Fan et al., 2020) and StrAttack (Xu et al., 2019), since our algorithm does not require multiple reruns of the whole algorithm with binary searched weights to ensure attack success and result quality.

GreedyFool (Dong et al., 2020), on the other hand, runs faster than our algorithm because of its greedy nature. However, previous experiments in Tables 1, 2, and 3 show that our algorithm always generate much sparser perturbations with lower ℓ_1 and ℓ_2 norms than Greedyfool under the same ℓ_∞ constraints.

Table 6. Empirical time cost of targeted attack in seconds.

Database	GreedyFool	Homotopy Attack	SAPF	StrAttack
CIFAR-10	0.43	22.64	100.15	142.65
ImageNet	71.49	516.57	1589.54	2242.78

3.8. Visualization

The visualizations for pixel-wise sparsity and group-wise sparsity are shown in Figure 1, and 2 respectively. In Figure 2, we use class activation map (CAM) (Zhou et al., 2016) to show whether the perturbations generated by our algorithm have a good correspondence with localized class-specific discriminative image regions. In the figures of perturbation positions, black pixels denote no perturbation; white pixels denote perturbing all three RGB channels; pure red, green, blue pixels represent perturbing single channel; and pixels with other color denote perturbing two channels.

From rows 2-6 of Figure 2, it is obvious that the generated perturbations cover the most discriminative areas of the objects, thanks to our algorithm achieving high group-sparsity. Row 1, on the other hand, might provide an interesting insight of the mechanism of DNN. It might indicate a high correlation between two different objects (trainer and killer whale) in the training data of DNN. The corresponding CAM also validates this finding.

The reason of perturbations generated with pixel-wise sparsity constraint not always being on discriminative areas like row 3 of Figure 1 is that, our algorithm is able to find some pixels not on the object that offer more adversary when enforcing pixel-wise constraint. However, when enforcing group-wise constraint, our algorithm will perturb groups with highest sum adversaries, which generally cover the discriminative areas.

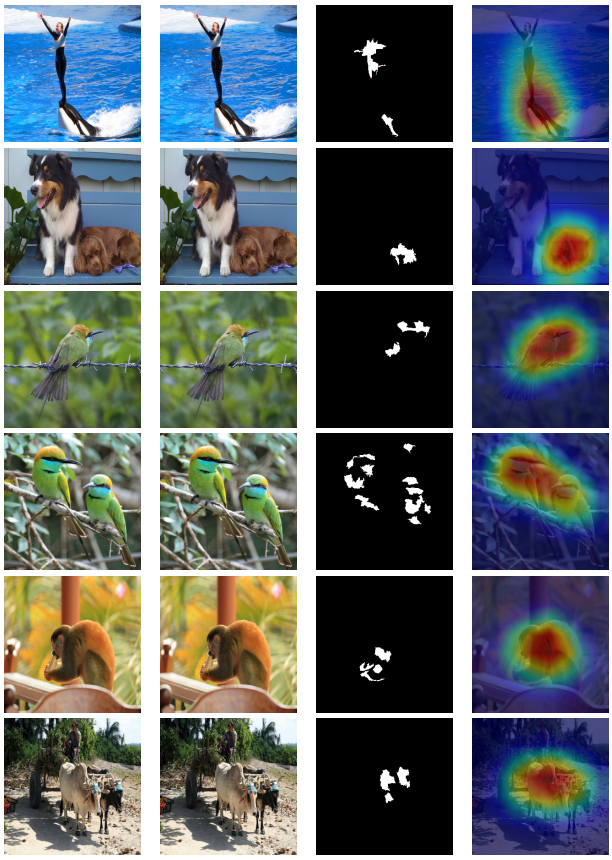

Figure 2. Visualization of targeted attack with group-wise sparsity. Each row from the left to right represents benign image, adversarial example, perturbation positions, and CAM of the original label. The ground-truth label for each row are killer whale, Sussex spaniel, bee eater, bee eater, titi monkey and oxcart, respectively. The target class is face powder for row 1 and 6, cock for row 2 and 3, and wild boar for row 4 and 5.

4. Conclusion

In this paper, we have proposed a novel homotopy algorithm for sparse adversarial attack based on Nonmonotone Accelerated Proximal Gradient Methods for Nonconvex Programming, an additional control of maximum ℓ_0 updates and an optional post attack stage per iteration. Extensive experiments show that our algorithm can generate very sparse adversarial perturbations while maintaining relatively low perturbation magnitudes, compared to the state-of-the-art methods. Also, our proposed control of maximum ℓ_0 updates and the optional post attack stage greatly improve the sparsity level of the homotopy algorithm.

References

Athalye, A., Engstrom, L., Ilyas, A., and Kwok, K. Synthesizing robust adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80, pp. 284–293, 2018.

Bach, F., Jenatton, R., Mairal, J., and Obozinski, G. Optimization with sparsity-inducing penalties. *Machine Learning*, 4(1):1–106, 2011.

Barzilai, J. and Borwein, J. M. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988.

Beck, A. and Hallak, N. Optimization problems involving group sparsity terms. *Mathematical Programming*, 178: 39–67, 2019.

Beck, A. and Teboulle, M. Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. *IEEE Transactions on Image Processing*, 18(11):2419–2434, 2009.

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

Croce, F. and Hein, M. Sparse and imperceivable adversarial attacks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4724–4732, 2019.

Deng, J., Dong, W., Socher, R., Li, L., Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, 2009.

Dong, X., Chen, D., Bao, J., Qin, C., Yuan, L., Zhang, W., Yu, N., and Chen, D. Greedyfool: Distortion-aware sparse adversarial attack. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pp. 4724–4732, 2020.

Du, X., Zhang, J., Han, B., Liu, T., Rong, Y., Niu, G., Huang, J., and Sugiyama, M. Learning diverse-structured networks for adversarial robustness. *arXiv preprint arXiv:2102.01886*, 2021.

Fan, Y., Wu, B., Li, T., Zhang, Y., Li, M., Li, Z., and Yang, Y. Sparse adversarial attack via perturbation factorization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 35–50, 2020.

Gao, R., Liu, F., Zhang, J., Han, B., Liu, T., Niu, G., and Sugiyama, M. Maximum mean discrepancy is aware of adversarial attacks. *arXiv preprint arXiv:2010.11415*, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Jain, P., Rao, N., and Dhillon, I. S. Structured sparse regression via greedy hard thresholding. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, volume 29, pp. 1516–1524, 2016.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, Department of Computer Science, University of Toronto, 2009.

Li, H. and Lin, Z. Accelerated proximal gradient methods for nonconvex programming. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, pp. 379–387, 2015.

Lin, Q. and Xiao, L. An adaptive accelerated proximal gradient method and its homotopy continuation for sparse optimization. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 73–81, 2014.

Lu, Z. Iterative hard thresholding methods for l_0 regularized convex cone programming. *Mathematical Programming*, 147(1-2):125–154, 2013.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

Modas, A., Moosavi-Dezfooli, S.-M., and Frossard, P. Sparsefool: A few pixels make a big difference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9087–9096, 2019.

Nesterov, Y. E. *Introductory Lectures on Convex Optimization - A Basic Course*, volume 87 of *Applied Optimization*. Springer, 2004.

Nguyen, A., Yosinski, J., and Clune, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 427–436, 2015.

Schott, L., Rauber, J., Bethge, M., and Brendel, W. Towards the first adversarially robust neural network model on mnist. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

Soussen, C., Idier, J., Duan, J., and Brie, D. Homotopy based algorithms for ℓ_0-regularized least-squares. *IEEE Transactions on Signal Processing*, 63(13):3301–3316, 2015.

Su, J., Vargas, D. V., and Sakurai, K. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016.

Wen, F., Chu, L., Liu, P., and Qiu, R. C. A survey on nonconvex regularization-based sparse and low-rank recovery in signal processing, statistics, and machine learning. *IEEE Access*, 6:69883–69906, 2018.

Xiao, C., Li, B., Zhu, J., He, W., Liu, M., and Song, D. Generating adversarial examples with adversarial networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3905–3911, 2018.

Xiao, L. and Zhang, T. A proximal-gradient homotopy method for the sparse least-squares problem. *SIAM Journal on Optimization*, 23(2):1062–1091, 2013.

Xu, K., Liu, S., Zhao, P., Chen, P.-Y., Zhang, H., Fan, Q., Erdogmus, D., Wang, Y., and Lin, X. Structured adversarial attack: Towards general implementation and better interpretability. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

Zhang, J., Xu, X., Han, B., Niu, G., Cui, L., Sugiyama, M., and Kankanhalli, M. Attacks which do not kill training make adversarial learning stronger. In *International Conference on Machine Learning*, pp. 11278–11287. PMLR, 2020a.

Zhang, J., Zhu, J., Niu, G., Han, B., Sugiyama, M., and Kankanhalli, M. Geometry-aware instance-reweighted adversarial training. *arXiv preprint arXiv:2010.01736*, 2020b.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2921–2929, 2016.

Zhou, H., Chen, D., Liao, J., Chen, K., Dong, X., Liu, K., Zhang, W., Hua, G., and Yu, N. Lg-gan: Label guided adversarial network for flexible targeted attack of point cloud based deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10356–10365, 2020.