A Sparse Hard-label Black-box Attack using Accelerated Proximal Gradient Methods

Satvik Mashkaria
Department of Computer Science
University of California, Los Angeles
satvikm@g.ucla.edu

Rohit Sunchu
Department of Computer Science
University of California, Los Angeles
sunchurohit@g.ucla.edu

Parth Shettiwar
Department of Computer Science
University of California, Los Angeles
parthshettiwar@g.ucla.edu

Abstract

Sparse adversarial attacks can fool deep neural networks (DNNs) by only perturbing a few pixels (regularized by l_0 norm). The resultant sparse and imperceptible attacks are practically relevant, and indicate an even higher vulnerability of DNNs that we usually imagined, as they pose a practical threat against real-world systems. 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. Moreover, such an attack leads to an NP-hard optimization problem. To make it more challenging, we restrict ourself to l_∞ imperceptibility on the perturbation magnitudes, while solving this optimzation function. We develop a novel proximal graident based algorithm, *SparseAPG*, based on homotopy algorithm of Schott et al. [2019], which works in the hard label black box untargetted label setup to solve the above optimization function and generate human-imperceptible adversarial sparse images. To the best of our knowledge, this is the first work in this setup. We compare our approach with existing similar baselines PointwiseSchott et al. [2018], SparseEvoVo et al. [2022], which don't account for l_∞ constraint. We perform experiments only on CIFAR-10 images with trained ResNet-18 model due to limited compute.

1 Introduction

Deep neural networks (DNNs) have been widely proved to be vulnerable to adversarial assaults, such as purposefully engineered minor perturbations to its inputs that can deceive them into making inaccurate and implausible predictions. Although unnoticeable to humans, these malevolent disturbances can elude and deceive DNNs. Thus, including DNNS into systems introduces a new attack surface as well as a motivation for malicious actors to target systems such as autonomous vehicles or machine learning models as a service (MLaaS) used in real-world applications such as self-driving automobiles. 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

In a black-box setting, an adversary may access all or only the top-1 predicted label and score—a score-based setting,—or simply the predicted label for a given input—a decision based setting. Importantly, the similarity measure, used to quantify the imperceptibility of the perturbation, can describe an attack as a dense attack, also know as l_2 constraint, or a l_∞ norm constrained adversarial attack or a sparse attack l_0 norm constrained adversarial attack. Though studied less, from a security standpoint, sparse attacks are particularly as threatening as dense attacks. One of the major reasons for this area getting less attention is it turning into an NP hard problem mainly due to the sparse l_0 constraint.

Furthermore, 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. Therefore, more recent methods focus on enforcing some element-wise magnitude constraints to also ensure imperceptible perturbations. Specifically, no work has been done yet in the black box hard label setup though, which becomes highly challenging with the above constraints but at same time most practical scenario in real world.

In short, our contributions from this work are:

- Inspired from Schott et al. [2019], which was done in white box setting, we propose a homotopy based algorithm, *SparseAPG* to perform hard-label black-box attack which minimizes the l_0 norm.
- Our method also bounds l_∞ norm while solving the objective, unlike the existing methods like Vo et al. [2022]. Also, our attack produces human imperceptible images at every iteration, unlike greedy methods like Schott et al. [2018].
- We shows results of our attack on CIFAR-10 images with trained ResNet-18 model (**?**), and compare the performance with the existing attacks in our setup.

2 Related Work

Many works have been done on generating sparse (l_0 norm bounded) attacks. We look at some of the methods in white-box setting as well as in black-box setting.

White-Box Attacks : Croce and Hein [2019] recently proposed a new method called PGD_0 that projects the adversarial perturbation produced by PGD Madry et al. [2017] to the l_0 ball. **?** introduced a Homotopy based Algorithm which focuses on minimizing the l_0 norm but also bounds the l_∞ norm at the same time. This method uses the nonmonotone Accelerated Proximal Gradient Method (nmAPG) for nonconvex programming followed by an l_0 change control step.

Hard Label Black-Box Attacks : In this setting, the adversary only has the top-1 predicted label of a neural network. This setting is the most restrictive and challenging scenario. Schott et al. [2019] introduced a new method called Pointwise Attack, a decision-based attack that greedily minimizes the l_0 norm. The attack first adds salt and pepper noise to an input image until it is misclassified. It then repeatedly iterates over all perturbed pixels, resetting them to the original pixel value if the perturbed image remains adversarial even after making this change. The attack ends when no pixel can be reset. Vo et al. [2022] proposed a evolutionary based algorithm called $SparseEvo$ to search for a desirable solution through an iterative process of improving upon potential solutions.

3 Problem Statement

In our sparse attack setting, we are given a normalized source image $x \in [0, 1]^{C\times W\times H}$ and its corresponding ground truth label y from the label set Y = $\{1, 2, \cdots, \text{K}\}$ where K denotes the number of classes, C, W and H denotes the number of channels, width and height of an image, respectively. The classifier that we aim to attack is $f : R^{C\times W\times H} \to Y$; our access is limited to its output label. In an untargeted setting, the adversary manipulates input x to change the decision of the classifier to any class label other than its ground-truth, i.e. $y' \in Y$ where $y' \neq y$. Formally, finding an adversarial attack on an image x_0 task can be formulated as a constrained optimization problem:

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

Where **1** is the indicator function. Optimizing this regularizer of l_0 norm of δ makes this optimization problem NP hard.

4 Method

In this section, we describe our method for the task described in the previous section. We start with a reformulation which casts the discrete problem into a continuous one. Next, we describe the preliminary Accelerated Proximal Gradient (APG) method for optimizing a non-convex non-differentiable objective function, and then move on to describe the nonmonotone extension which is used in our method. Finally, we describe SparseAPG, which is built on the top of nonmonotone APG.

4.1 Reformulation

Directly optimizing over the problem 1 is very difficult because of the discrete nature of the function. Instead of optimizing directly on the adversarial image, following the idea from Cheng et al. [2018], we try to optimize over the direction along which we have to move minimum to achieve an adversarial image. Formally, we define $g : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}$ as

$$g(\boldsymbol{\theta}) = \text{argmin}_{\lambda > 0} \left\{ f(x_0 + \lambda \frac{\|\boldsymbol{\theta}\|}{\boldsymbol{\theta}}) \neq y_0 \right\} \tag{2}$$

Intuitively, this $g(\boldsymbol{\theta})$ gives the minimum distance that you have to move along the direction of $\boldsymbol{\theta}$ to obtain an adversarial image. Thus, our problem now reduces to finding direction which minimizes g. Now, g becomes close to a continuous function which is possible to optimize. Moreover, since we are interested in finding a sparse solution, we add L_0 norm to our objective. Hence, our optimization objective for a benign image $\mathbf{x}_0$ is now becomes

$$\begin{aligned} &\underset{\boldsymbol{\theta}}{\text{argmin}}\; G(\boldsymbol{\theta}) \\ &\text{s.t. } \|\boldsymbol{\theta}\|_\infty < \epsilon \text{ and } 0 < \mathbf{x}_0 + \boldsymbol{\theta} < 1, \\ &\text{ where } G(\boldsymbol{\theta}) = g(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_0 \end{aligned} \tag{3}$$

Note that although we can assume g to be differentiable, G is not differentiable because of the L_0 norm, which is not differentiable.

4.2 Gradient Calculations

In this subsection, we describe the gradient calculations for $g(\mathbf{x})$. Since we don't have an analytical solution for the gradient of 2, we have to approximate it using numerical methods. A naive method would be to directly use the first principle.

$$\frac{\partial g(\mathbf{x})}{\partial \mathbf{x}_{ij}} = \frac{g(\mathbf{x}^h) - g(\mathbf{x}^l)}{2\epsilon} \tag{4}$$

where $\mathbf{x}^h$ (and $\mathbf{x}^l$) are obtained by adding (and subtracting) ϵ from $(i, j)^{th}$ pixel of $\mathbf{x}$, keeping everything else the same. Formally, $\mathbf{x}^h_{ij} = \mathbf{x}_{ij} + \epsilon$, and $\mathbf{x}^h_{ab} = \mathbf{x}_{ab} + \epsilon$, $\forall a \neq i$ and $b \neq j$, similar for $\mathbf{x}^l$. However, this method is extremely slow, because we have to calculate gradient for every pixel. For CIFAR-10 images this method will make about $60,000$ forward calls to the model, which is very slow, and practically impossible to use.

Instead of using precise gradient, we use approximate gradients which are easy to compute. Specifically, we use Randomized Gradient Free (RGF) approach to approximate gradients Cheng et al. [2018]. It is known that

$$g'(\mathbf{x}) = \underset{\mathbf{u} \sim \mathcal{N}(\mathbf{0}, I)}{\mathbb{E}} \frac{g(\mathbf{x} + \epsilon * \mathbf{u}) - g(\mathbf{u})}{\epsilon} \mathbf{u} \tag{5}$$

Hence, we approximate gradient as

$$\hat{g'}(\mathbf{x}) = \frac{1}{Q} \sum_{q=1}^{Q} \frac{g(\mathbf{x} + \epsilon * \mathbf{u_q}) - g(\mathbf{x})}{\epsilon} \mathbf{u_q} \tag{6}$$

where $u_q \sim \mathcal{N}(\mathbf{0}, I), \forall q \in \{1, \cdots, Q\}$. We typically use $Q = 40$ and $\epsilon = 0.005$ in our experiments. This method reduces the number of forward calls to 800.

4.3 Accelerated Proximal Gradient (APG)

This algorithm, proposed by Li and Lin [2015], is used to optimize functions which take the form $F(x) = f(x) + g(x)$, where f is a Lipchitz continuous function (can be non-convex) and g is any function, which can be both non-convex and non-differentiable. This is a challenging problem to solve because the gradients are not directly available.

Algorithm 1 describes this algorithm. Here $\text{prox}_{\alpha g}(\mathbf{x}) = \text{argmin}_{\mathbf{u}} g(\mathbf{u}) + \frac{1}{2\alpha}\|\mathbf{x} - \mathbf{u}\|$ is a proximal gradient of F obtained by a second order approximation of F, and $\alpha = \frac{1}{L}$, where L is the Liptchitz constant of f Beck and Teboulle [2009]. This update rule is similar to Nesterov Accelerated Gradient Descent rule, where we first move in the direction of momentum and then make the corrective gradient update. For mroe details, refer to Li and Lin [2015].

Algorithm 1 monotone APG with fixed stepsize

```
Initialize $\mathbf{z}_1 = \mathbf{x}_1 = \mathbf{x}_0$, $t_1 = 1$, $t_0 = 0$, $\alpha_y < \frac{1}{L}$, $\alpha_x < \frac{1}{L}$.
for $k = 1, 2, 3, \cdots$ do
    $$\mathbf{y}_k = \mathbf{x}_k + \frac{t_{k-1}}{t_k}(\mathbf{z}_k - \mathbf{x}_k) + \frac{t_{k-1} - 1}{t_k}(\mathbf{x}_k - \mathbf{x}_{k-1}), \quad (13)$$
    $$\mathbf{z}_{k+1} = \text{prox}_{\alpha_y g}(\mathbf{y}_k - \alpha_y \nabla f(\mathbf{y}_k)), \quad (14)$$
    $$\mathbf{v}_{k+1} = \text{prox}_{\alpha_x g}(\mathbf{x}_k - \alpha_x \nabla f(\mathbf{x}_k)), \quad (15)$$
    $$t_{k+1} = \frac{\sqrt{4(t_k)^2 + 1} + 1}{2}, \quad (16)$$
    $$\mathbf{x}_{k+1} = \begin{cases} \mathbf{z}_{k+1}, & \text{if } F(\mathbf{z}_{k+1}) \leq F(\mathbf{v}_{k+1}), \\ \mathbf{v}_{k+1}, & \text{otherwise.} \end{cases} \quad (17)$$
end for
```

4.4 nonmonotone APG (nmAPG)

Note that the algorithm described in previous section is monotonic, i.e. function value either remains same or decreases at every iteration. However, this kind of monotonicity is not desired when the function is highly non-convex. This is because if we are stuck in a bad local minima or valley, there is no way we can come out. Therefore, Li and Lin [2015] proposes a nonmonotone version of APG, which relaxes the criteria of monotonicity to "push" the solution out of bad local minima regions.

nmAPG is described in Algorithm 3. Here c_k is a relaxation over $F(x_k)$ and the algorithm ensure sufficient descent from c_k instead of $F(x_k)$. At each iteration k, c_k is obtained by exponential smoothing of previous iteration function values.

4.5 Additions over nmAPG

We augment the nmAPG algorithm by the following three ways to optimize for l_0 norm.

4.5.1 Initial Lambda Search

Value of λ plays a crucial role in the convergence and sparsity of the solution. In such scenario, setting an ad-hoc value for λ might not give good solution for all the cases. Thus, motivated from Zhu et al. [2021], we perform a line search over the initial value of λ. In the first coarse search step, we keep increasing the λ until the first iteration of nmAPG algorithm updates the solution. Then in the fine search step, we keep decreases the value by a loose factor until the updates stop.

Algorithm 3 nonmonotone APG with fixed stepsize

```
Initialize $\mathbf{z}_1 = \mathbf{x}_1 = \mathbf{x}_0$, $t_1 = 1$, $t_0 = 0$, $\eta \in [0, 1)$, $\delta > 0$, $c_1 = F(\mathbf{x}_1)$, $q_1 = 1$, $\alpha_x < \frac{1}{L}$, $\alpha_y < \frac{1}{L}$.
for $k = 1, 2, 3, \cdots$ do
    $\mathbf{y}_k = \mathbf{x}_k + \frac{t_{k-1}}{t_k}(\mathbf{z}_k - \mathbf{x}_k) + \frac{t_{k-1} - 1}{t_k}(\mathbf{x}_k - \mathbf{x}_{k-1})$,    (140)
    $\mathbf{z}_{k+1} = \text{prox}_{\alpha_y g}(\mathbf{y}_k - \alpha_y \nabla f(\mathbf{y}_k))$,    (141)
    if $F(\mathbf{z}_{k+1}) \leq c_k - \delta \|\mathbf{z}_{k+1} - \mathbf{y}_k\|^2$ then
        $\mathbf{x}_{k+1} = \mathbf{z}_{k+1}$.    (142)
    else
        $\mathbf{v}_{k+1} = \text{prox}_{\alpha_x g}(\mathbf{x}_k - \alpha_x \nabla f(\mathbf{x}_k))$,    (143)
        $\mathbf{x}_{k+1} = \begin{cases} \mathbf{z}_{k+1}, & \text{if } F(\mathbf{z}_{k+1}) \leq F(\mathbf{v}_{k+1}), \\ \mathbf{v}_{k+1}, & \text{otherwise.} \end{cases}$    (144)
    end if
    $t_{k+1} = \frac{\sqrt{4(t_k)^2 + 1} + 1}{2}$,    (145)
    $q_{k+1} = \eta q_k + 1$,    (146)
    $c_{k+1} = \frac{\eta q_k c_k + F(\mathbf{x}_{k+1})}{q_{k+1}}$.    (147)
end for
```

4.5.2 Additional Sparsity Control

After each inner iteration of nmAPG, we choose the top v non-zero values to ensure sparsity at every step. v is a hyper-parameter for the algorithm. For more details, refer Zhu et al. [2021].

4.6 Masked Gaussian Initialization

Note that we cannot initialize our algorithm with $x_0 = \mathbf{0}$, because we are working with directions and $g(\mathbf{0}) \to \infty$. Instead, we sample a zero-mean Gaussian x_g with a small variance (we use $\sigma = 0.01$), and then apply a binary mask M over x_g to obtain the initial value x_0. Each element of the mask M is sampled from a Bernoulli distribution with $p = 0.05$, i.e., $M_{ij} \sim \text{Bernoulli}(0.05), \forall i, j$. This is to ensure sparsity in the initialization of the algorithm, which is crucial in the smooth convergence of the algorithm.

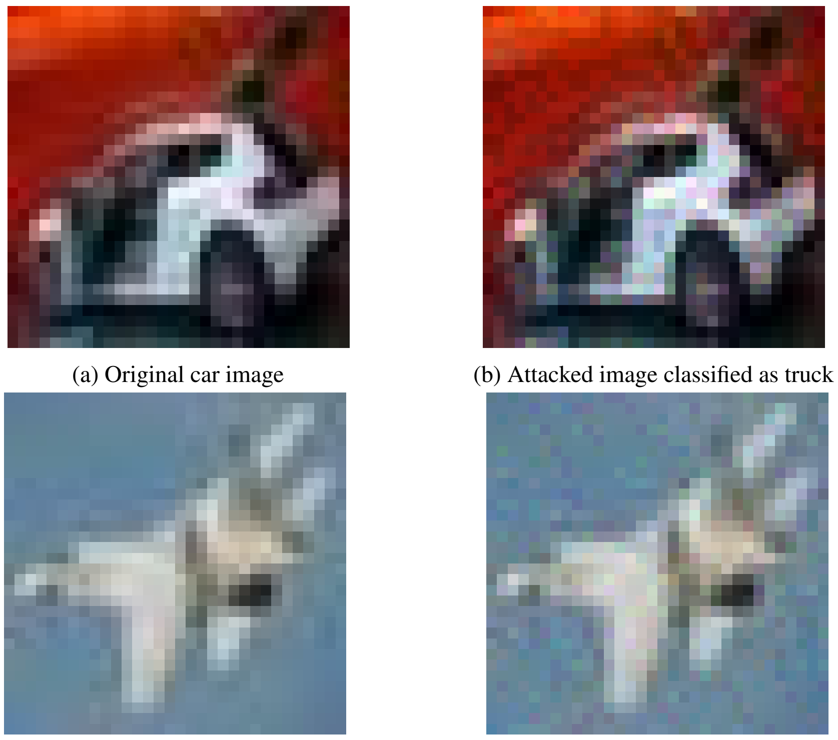
5 Experiments and Results

We perform experiments on CIFAR-10 Krizhevsky et al. images with a trained ResNet-18 model He et al. [2015] which achieves $\sim 85\%$ test accuracy. In the Table 1, we compare the l_0 and l_∞ norms of 3 attacks - Pointwise, SparseEvo and SparseAPG (our algorithm).

Attack	l_0 norm	l_∞ norm
Pointwise	25	0.627
SparseEvo	18	0.673
SparseAPG	507	0.05

Table 1: Comparison of l_0 and l_∞ norms of various attacks, averaged over 10 images. Note that our l_0 norm currently is not good but our l_∞ norm is bounded by the provided value $\epsilon = 0.05$. Note that success rate for all the attacks is 100%.

```
input $x_0$: benign image; $t$: target; $\epsilon$: invisible threshold;
    $eta$, $delta$, $rho$, MaxIter: parameters of nmAPG;
    $c$, $v$, $\beta$, $\gamma$: parameters of this algorithm.
output $\delta$.
1: $\delta^0 = 0$; $v_{ini} = v$; $k = 0$;
2: $\lambda$ =Lambda_Search($x_0$, $t$, $c$, $eta$, $delta$, $rho$, $v$, $\beta$);
3: repeat
4:     $\delta^{k+1}$ = nmAPG($\delta^k$, $x_0$, $t$, $\lambda$, $eta$, $delta$, $rho$, $v$, MaxIter);
5:     $v = v_{ini}$;
6:     if not success then
7:         if $||\delta^{k+1}||_1 \le ||\delta^{k+1}||_0 * \epsilon * \gamma$ then
8:             Set $v$ as a small integer;
10:        end if
11:        Decrease $\lambda$ by a factor;
12:    end if
13:    $k = k + 1$;
14: until Attack Succeeds.
```


(a) Original car image (b) Attacked image classified as truck

(c) Original Aeroplane image (d) Attacked image classified as cat

Figure 1: Adversarial examples of our model. Notice that the l_∞ norm of the adversarial image is bounded by $\epsilon = 0.05$, hence individual pixels doesn't change much.

From the table, we see that Pointwise and SparseEvo attacks do not bound the l_∞ norm, hence they have very low l_0 norm. Whereas, SparseAPG constrains the l_∞ norm to be less than 0.05. Hence, SparseAPG requires more pixels to be perturbed compared to Pointwise and SparseEvo. We believe that we can further improve the l_0 norm by proper parameter tuning and modifications.

6 Known issues and possible solutions

In this section, we describe some of the possible problems we suspect with our current version of the algorithm and their possible solutions.

1. Sometime when the current solution in an iteration is very sparse, the g value for that direction becomes very large (or even unbounded in some cases), which creates problem with convergence. A possible solution for this problem would be to perturb a few more pixels following some simple heuristic to escape from such a situation.
2. Our method is currently slow. For the gradient calculations, we perform sequential model queries. We can make it parallelized by batching independent operations to make it fast.

7 Conclusion and Future Work

Very less work has been done for sparse hard-label black-box attacks with bounded l_∞ norm. We propose an algorithm based on Accelerated Proximal Gradient methods to solve this problem posed as an optimization task. Our method is currently in an early phase and there is a large scope of improvement. As future work, we plan to improve the performance of the method by improving upon some of the problems we discussed in the previous section.

For 1 seocnd bucket, ignore adds whose px is far from fair_px by more than 6 bips. 6 bips is the 99.9^{th} %-ile of px - fair_px relative difference in bips. For remaining adds,

$$\text{alpha\_sqrt\_px\_diff} = \frac{\sum_i \left(\sqrt{\text{qty}_i} * \left(\frac{\text{fair\_px}_\text{i} - \text{px}_\text{i}}{\text{fair\_px}_\text{i}}\right)\right)}{\sum_i \sqrt{\text{qty}_i}} \tag{7}$$

where qty is $notional\_qty$ clipped by 80% expanding quantile value.

$$\text{BV} = \sum_i \left(\text{qty}_i * \exp\left(-c\frac{\text{fair\_px}_\text{i} - \text{px}_\text{i}}{\text{decayed\_spread}}\right)\right) \tag{8}$$

References

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009. doi: 10.1137/080716542. URL `https://doi.org/10.1137/080716542`.

Minhao Cheng, Thong Le, Pin-Yu Chen, Jinfeng Yi, Huan Zhang, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack:an optimization-based approach, 2018. URL `https://arxiv.org/abs/1807.04457`.

Francesco Croce and Matthias Hein. Sparse and imperceivable adversarial attacks, 2019. URL `https://arxiv.org/abs/1909.05040`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL `https://arxiv.org/abs/1512.03385`.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL `http://www.cs.toronto.edu/~kriz/cifar.html`.

Huan Li and Zhouchen Lin. Accelerated proximal gradient methods for nonconvex programming. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL `https://proceedings.neurips.cc/paper/2015/file/f7664060cc52bc6f3d620bcedc94a4b6-Paper.pdf`.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2017. URL `https://arxiv.org/abs/1706.06083`.

Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. Towards the first adversarially robust neural network model on mnist, 2018. URL `https://arxiv.org/abs/1805.09190`.

Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. Towards the first adversarially robust neural network model on MNIST. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=S1EHOsC9tX`.

Viet Quoc Vo, Ehsan Abbasnejad, and Damith C. Ranasinghe. Query efficient decision based sparse attacks against black-box deep learning models. *CoRR*, abs/2202.00091, 2022. URL `https://arxiv.org/abs/2202.00091`.

Mingkang Zhu, Tianlong Chen, and Zhangyang Wang. Sparse and imperceptible adversarial attack via a homotopy algorithm, 2021. URL `https://arxiv.org/abs/2106.06027`.