Sparse Adversarial Attack via Perturbation Factorization

Yanbo Fan[1†], Baoyuan Wu[1†*], Tuanhui Li[1], Yong Zhang[1], Mingyang Li[2], Zhifeng Li[1], Yujiu Yang[2]

[1] Tencent AI Lab
[2] Tsinghua Shenzhen International Graduate School, Tsinghua University
wubaoyuan1987@gmail.com

Abstract. This work studies the sparse adversarial attack, which aims to generate adversarial perturbations onto partial positions of one benign image, such that the perturbed image is incorrectly predicted by one deep neural network (DNN) model. The sparse adversarial attack involves two challenges, *i.e.*, where to perturb, and how to determine the perturbation magnitude. Many existing works determined the perturbed positions manually or heuristically, and then optimized the magnitude using a proper algorithm designed for the dense adversarial attack. In this work, we propose to factorize the perturbation at each pixel to the product of two variables, including the perturbation magnitude and one binary selection factor (*i.e.*, 0 or 1). One pixel is perturbed if its selection factor is 1, otherwise not perturbed. Based on this factorization, we formulate the sparse attack problem as a mixed integer programming (MIP) to jointly optimize the binary selection factors and continuous perturbation magnitudes of all pixels, with a cardinality constraint on selection factors to explicitly control the degree of sparsity. Besides, the perturbation factorization provides the extra flexibility to incorporate other meaningful constraints on selection factors or magnitudes to achieve some desired performance, such as the group-wise sparsity or the enhanced visual imperceptibility. We develop an efficient algorithm by equivalently reformulating the MIP problem as a continuous optimization problem. Extensive experiments demonstrate the superiority of the proposed method over several state-of-the-art sparse attack methods. The implementation of the proposed method is available at https://github.com/wubaoyuan/Sparse-Adversarial-Attack.

Keywords: Perturbation Factorization, Sparse Adversarial Attack, Mixed Integer Programming

1 Introduction

Deep neural networks (DNNs) have achieved a great success in many applications, such as image classification [16,32,35], face recognition [31,34], natural

† indicates equal contribution. ∗ indicates corresponding author.

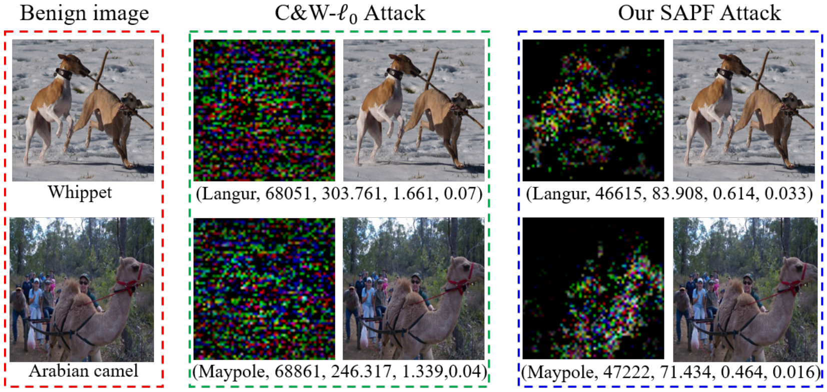
Benign image
C&W-ℓ_0 Attack
Our SAPF Attack
Whippet
(Langur, 68051, 303.761, 1.661, 0.07)
(Langur, 46615, 83.908, 0.614, 0.033)
Arabian camel
(Maypole, 68861, 246.317, 1.339,0.04)
(Maypole, 47222, 71.434, 0.464, 0.016)

Fig. 1: Examples of the targeted sparse adversarial attack to image classification on two benign images selected from ImageNet [11]. (**Left**): benign images with their ground-truth labels below. (**Middle**): perturbations generated by the C&W-ℓ_0 attack [7]. (**Right**): perturbations generated by our attack method. The text under each perturbation indicates the target attack class, $\ell_0, \ell_1, \ell_2, \ell_\infty$-norms of the perturbation, respectively. Our method successfully attacks the benign image to the target class with fewer perturbed pixels and lower distortion compared to the C&W-ℓ_0 attack.

language processing [29], etc. However, it is discovered that DNNs are vulnerable to adversarial examples [2,15,33], where small malicious perturbations can cause DNNs to make incorrect predictions. It has been observed in many DNN based applications, such as image classification [14,9,21], image captioning [39,8], image retrieval [13,4], question answering [23], autonomous driving [24], automatic check-out [25], face recognition [12], face detection [22], *etc.*.

Most existing methods of adversarial attacks focus on optimizing the magnitudes of perturbations such that the perturbations are imperceptible to human eyes, while the perturbed positions are not considered as assuming that all pixels will be perturbed. It is called the *dense adversarial attack*. In contrast, some recent works [17,7] observed that the DNN model can also be fooled if only partial positions (even one pixel of one image [30]) are perturbed, dubbed *sparse adversarial attack*. Compared to the dense attack, as analyzed in the previous work [38], the sparse attack not only produces fewer perturbations, but also provides additional insights about the vulnerability of DNNs, *i.e.*, a better interpretation of adversarial attacks. For example, as shown in Fig. 1, the sparse perturbations generated by our attack method mainly occur on the positions of the main object in the benign image, such as the body area of "Arabian camel" in the second benign image. The perturbation positions reveal that which part of one image is important but also fragile for its prediction by the DNN model. Despite these advantages, there is also one additional challenge for the sparse attack, *i.e.*, how to determine the perturbed positions. Some existing works (*e.g.*, LaVAN [17]) manually determined a local patch. Then the attack algorithm designed for the dense attack is adopted to generate perturbations within this local patch. Some works tried to determine the perturbed positions using heuristic strategies. For example, C&W-ℓ_0 [7] gradually fixed some pixels that don't contribute much to the classification output in each iteration. There is no guarantee that these heuristic methods could identify satisfied perturbed positions.

In this work, we provide a new perspective that the perturbation at each single pixel can be factorized according to its two characteristics, *i.e.*, magnitude and position. Consequently, each single perturbation can be represented by the product between the perturbation magnitude and a binary selection factor (*i.e.*, 0 or 1). If the selection factor is 1, then the corresponding pixel is perturbed, otherwise not perturbed. This simple perspective brings in multiple benefits. **First**, the sparse adversarial attack can be formulated as a mixed integer programming (MIP) problem, which jointly optimizes the binary selection factors and the continuous perturbation magnitudes of all pixels in one image. And, the number of perturbed pixels can be explicitly enforced by imposing a cardinality constraint on all binary selection factors. In contrast to aforementioned two-state methods (*e.g.*, [17,7]), the proposed joint optimization is expected to generate better perturbations (*i.e.*, fewer perturbed positions or smaller perturbation magnitudes), and enables the more convenient control of the degree of sparsity. **Second**, the perturbation factorization provides the extra flexibility to impose some meaningful constraints on the perturbation magnitude or the binary selection factor, to achieve some desired attack performance. We present two case studies. One is *group sparsity* on the selection factors to encourage the perturbations to occur together. The other is introducing a *prior weight* of each pixel onto the perturbation magnitude, according to the pixel values of the benign image, to enhance the imperceptibility to human visual perception. Both of them can be naturally embedded into the proposed joint optimization problem. Moreover, the MIP problem is NP-hard and cannot be directly optimized by any off-the-shelf continuous optimization solver. Inspired by one recent method called ℓ_p-Box ADMM [36] designed for integer programming (IP), we propose to reformulate the MIP problem to an equivalent continuous optimization problem, which is then efficiently solved using an iterative scheme. Finally, we conduct extensive experiments on two benchmark databases, including CIFAR-10 [18] and ImageNet [11], to verify the performance of the proposed method.

The main contributions of this work are four-fold. **1)** We provide a new perspective that the perturbation on each pixel can be factorized as the product between the perturbation magnitude and one binary selection factor. **2)** We formulate the sparse adversarial attack as a MIP problem to jointly optimize perturbation magnitudes and binary selection factors, with a cardinality constraint on selection factors to exactly control the sparsity. **3)** We develop an effective and efficient continuous algorithm to solve the MIP problem. **4)** Experimental results on two benchmark databases demonstrate that the proposed model is superior to several state-of-the-art sparse attack methods.

2 Related Work

In this section, we focus on existing works of sparse adversarial attacks. In contrast to dense adversarial attacks, one special challenge in sparse adversarial attacks is how to determine perturbed positions. According to the strategies for tackling this challenge, we categorize existing sparse attack methods into three types, including *manual, heuristic and optimized* strategies. **First**, the manual strategy means that the attacker manually specify the perturbed positions. For example, LaVAN [17] proposed to add a adversarial but visible local patch onto the benign image to fool the CNN-based classification model. It demonstrated that the model may be fooled by a small patch (about 2% of the image) located at the background region. However, the exact position of the local patch is manually determined by the attacker. **Second**, the perturbed positions are determined following some heuristic strategies in some works. For example, the method called Jacobian-based Saliency Map Attack (JSMA) [28] and its extensions [7] proposed to determine the perturbed positions according to the saliency map. CornerSearch [10] utilized a heuristic sampling to determine the perturbed pixels. **Third**, some works attempted to optimize the perturbed positions. For example, the One-Pixel attack [30] explored the extreme case that only one pixel is attacked to fool the DNN model. The perturbed pixel is searched using the differential evolution (DE) algorithm. Another attempt proposed in [41] utilized the ℓ_0 minimization to enforce the sparsity of perturbations. The alternating direction method of multipliers (ADMM) method is then adopted to separate the ℓ_0-norm and the adversarial loss, to facilitate the optimization of the sparse attack. However, there is no constraint on perturbation magnitudes, leading to that the learned perturbation may be very large to be visible. Besides, since the sparsity is enforced via the ℓ_0 term in the objective function, it is difficult to exactly control the degree of sparsity. Apart from the pixel-wise sparsity, [38] investigated the group-wise sparsity in adversarial attacks, motivated by the group Lasso [40]. They showed that group-wise sparsity property provides the better interpretability for adversarial attack and demonstrated that the learned perturbation is highly related to discriminative image regions. The group-wise sparsity can be naturally embedded into our sparse adversarial attack model.

3 Sparse Adversarial Attack

3.1 Preliminaries of Adversarial Attack

We denote the classification model as $f : \mathcal{X} \to \mathcal{Y}$, with $\mathcal{X} \in [0,1]^{w \times h \times c}$ being the image space and $\mathcal{Y} = \{1, \ldots, K\}$ being the K-class output space. $\mathbf{x} \in \mathcal{X}$ indicates one benign image, and $y \in \mathcal{Y}$ is its ground-truth label. $f(\mathbf{x}) \in [0,1]^K$ denotes the posterior vector. The adversarial attack is generally formulated as

$$\min_{\boldsymbol{\epsilon}} \|\boldsymbol{\epsilon}\|_p^p + \lambda \mathcal{L}\big(f(\mathbf{x} + \boldsymbol{\epsilon}), y_t\big), \text{ s.t. } \mathbf{x} + \boldsymbol{\epsilon} \in [0,1], \tag{1}$$

where the loss function $\mathcal{L}$ is specified according to y_t: if $y_t = y$, then it is called the untargeted attack, and $\mathcal{L}$ is set as the negative cross entropy function; if $y_t \neq y$

is another target label assigned by the attack, then it is called the targeted attack, and $\mathcal{L}$ is set as the cross entropy function. In this work, we focus on the targeted attack, since it is more challenging than the untargeted attack. The value p could be specified as different values, according to the attacker's requirement. The widely used values include $p = 2$ (*e.g.*, C&W-L2 [7]), $p = \infty$ (*e.g.*, FGSM [33]). Using these norms, the adversarial perturbations could be added at all pixels, dubbed *dense attack*. In contrast, if $p = 0$, then the above problem will encourage that only a few pixels are perturbed, dubbed *sparse attack*. However, it is difficult to directly optimize the above problem, due to the non-differentiability of ℓ_0-norm. Instead, some existing works (*e.g.*, [17,7]) proposed to alternatively determine the perturbed positions using some heuristic strategies and optimiz the magnitudes of perturbations. In contrast, we propose to optimize the perturbed positions and the perturbation magnitudes jointly, as specified below. For clarity, $\mathbf{x}$ and $\boldsymbol{\epsilon}$ are reshaped from the tensor to the vector, *i.e.*, $\mathbf{x}, \boldsymbol{\epsilon} \in \mathbb{R}^N$, with $N = w \cdot h \cdot c$.

3.2 Sparse Adversarial Attack via Perturbation Factorization

We firstly factorize the perturbation $\boldsymbol{\epsilon}$ as follows:

$$\boldsymbol{\epsilon} = \boldsymbol{\delta} \odot \mathbf{G}, \tag{2}$$

where $\boldsymbol{\delta} \in \mathbb{R}^N$ denotes the vector of perturbation magnitudes; $\mathbf{G} \in \{0,1\}^N$ denotes the vector of perturbed positions; $\odot$ represents element-wise product. Utilizing this factorization, we propose a new formulation of the sparse adversarial attack, as follows:

$$\min_{\boldsymbol{\delta},\mathbf{G}} \; \|\boldsymbol{\delta} \odot \mathbf{G}\|_2^2 + \lambda_1 \mathcal{L}(f(\mathbf{x} + \boldsymbol{\delta} \odot \mathbf{G}), y_t), \quad \text{s.t.} \;\; \mathbf{1}^\top \mathbf{G} = k, \;\; \mathbf{G} \in \{0,1\}^N, \tag{3}$$

where $\lambda_1 > 0$ is a trade-off parameter. The cardinality constraint $\mathbf{1}^\top \mathbf{G} = k$ is introduced to enforce that only $k < N$ pixels are perturbed. Note that the range constraint $\mathbf{x} + \boldsymbol{\delta} \odot \mathbf{G}$ is omitted here, as it can be simply satisfied via clipping. Since $\boldsymbol{\delta}$ is continuous, while $\mathbf{G}$ is integer, Problem (3) is a mixed integer programming (MIP) problem. Problem (3) is denoted as **SAPF** (Sparse adversarial Attack via Perturbation Factorization).

3.3 Continuous Optimization for the MIP Problem

The mixed integer programming (MIP) problem is challenging, as it cannot be directly optimized using any off-the-shelf continuous solver. Recently, a generic method for integer programming called ℓ_p-Box ADMM [36] proved that the discrete constraint space can be equivalently replaced by the intersection of two continuous constraints, and it showed very superior performance in many integer programming tasks, such as image segmentation, matching, clustering [5], MAP inference [37], model compression for CNNs [20], *etc.*. Inspired by that,

Algorithm 1 Continuous optimization for the MIP problem (5).

```
Input: benign image $\{\mathbf{x}, y_0\}$, target attack class $y_t$, number of perturbed pixels $k$ and
    trade-off parameter $\lambda_1$.
Output: $\mathbf{x} + \boldsymbol{\delta} \odot \mathbf{G}$.
Initialize $\mathbf{G} = \mathbf{1}$ and $\boldsymbol{\delta} = \mathbf{0}$.
while not converged do
    Given $\mathbf{G}$, update $\boldsymbol{\delta}$ with gradient descent (see Step 1 and sub-problem (6) );
    Given $\boldsymbol{\delta}$, update $\mathbf{G}$ with ADMM (see Step 2 and sub-problem (8)).
end while
```

we propose to equivalently reformulate the MIP problem to a continuous optimization problem, which is then efficiently optimized via an iterative scheme. Specifically, the binary constraints on $\mathbf{G}$ could be replaced as follows:

$$\mathbf{G} \in \{0,1\}^N \Leftrightarrow \mathbf{G} \in \mathcal{S}_b \cap \mathcal{S}_p, \tag{4}$$

where $\mathcal{S}_b = [0,1]^N$ is a box constraint and $\mathcal{S}_p = \left\{\mathbf{G} : \|\mathbf{G} - \frac{1}{2}\|_2^2 = \frac{N}{4}\right\}$ is an ℓ_2-sphere constraint. Due to the space limit, we refer the reader to [36] for the detailed proof of (4). Utilizing (4), Problem (3) is equivalently reformulated as

$$\begin{aligned} \min_{\boldsymbol{\delta}, \mathbf{G}, \mathbf{Y}_1 \in \mathcal{S}_b, \mathbf{Y}_2 \in \mathcal{S}_p} & \|\boldsymbol{\delta} \odot \mathbf{G}\|_2^2 + \lambda_1 \mathcal{L}(f(\mathbf{x} + \boldsymbol{\delta} \odot \mathbf{G}), y_t) \\ \text{s.t.} \quad & \mathbf{1}^\top \mathbf{G} = k, \ \mathbf{G} = \mathbf{Y}_1, \ \mathbf{G} = \mathbf{Y}_2, \end{aligned} \tag{5}$$

where $\mathbf{Y}_1$ and $\mathbf{Y}_2$ are two additional variables to decompose the box and the ℓ_2-sphere constraints on $\mathbf{G}$. Due to the element-wise product between $\boldsymbol{\delta}$ and $\mathbf{G}$, they should be alternatively optimized. The general structure of the optimization is summarized in Algorithm 1, of which details are shown below.

Step 1: Given $\mathbf{G}$, Update $\boldsymbol{\delta}$ by Gradient Descent. Given $\mathbf{G}$, the sub-problem w.r.t. $\boldsymbol{\delta}$ is as follows:

$$\min_{\boldsymbol{\delta}} \|\boldsymbol{\delta} \odot \mathbf{G}\|_2^2 + \lambda_1 \mathcal{L}(f(\mathbf{x} + \boldsymbol{\delta} \odot \mathbf{G}), y_t). \tag{6}$$

It is very similar to the formulation of the dense adversarial attack, and can be solved by the gradient descent algorithm, as follows:

$$\boldsymbol{\delta} \leftarrow \boldsymbol{\delta} - \eta_{\boldsymbol{\delta}} \cdot \nabla \boldsymbol{\delta} = \boldsymbol{\delta} - \eta_{\boldsymbol{\delta}} \cdot \left[2(\boldsymbol{\delta} \odot \mathbf{G} \odot \mathbf{G}) + \lambda_1 \frac{\partial \mathcal{L}(f(\mathbf{x} + \boldsymbol{\delta} \odot \mathbf{G}), y_t)}{\partial \boldsymbol{\delta}} \right], \tag{7}$$

where $\eta_{\boldsymbol{\delta}} > 0$ and the number of gradient steps will be specified in experiments.

Step 2: Given $\boldsymbol{\delta}$, Update $\mathbf{G}$ using ADMM. Given $\boldsymbol{\delta}$, the sub-problem w.r.t. $(\mathbf{G}, \mathbf{Y}_1, \mathbf{Y}_2)$ is as follows:

$$\begin{aligned} \min_{\mathbf{G}, \mathbf{Y}_1 \in \mathcal{S}_b, \mathbf{Y}_2 \in \mathcal{S}_p} & \|\boldsymbol{\delta} \odot \mathbf{G}\|_2^2 + \lambda_1 \mathcal{L}(f(\mathbf{x} + \boldsymbol{\delta} \odot \mathbf{G}), y_t) \\ \text{s.t.} \quad & \mathbf{1}^\top \mathbf{G} = k, \ \mathbf{G} = \mathbf{Y}_1, \ \mathbf{G} = \mathbf{Y}_2. \end{aligned} \tag{8}$$

It can be optimized by the alternating direction method of multipliers (ADMM) algorithm [6]. Specifically, the augmented Lagrangian function of (8) is

$$
\begin{aligned}
&L(\mathbf{G}, \mathbf{Y}_1, \mathbf{Y}_2, \mathbf{Z}_1, \mathbf{Z}_2, z_3) = \|\boldsymbol{\delta} \odot \mathbf{G}\|_2^2 + \lambda_1 \mathcal{L}(f(\mathbf{x} + \boldsymbol{\delta} \odot \mathbf{G}), y_t) + (\mathbf{Z}_1)^\top (\mathbf{G} - \mathbf{Y}_1) \\
&+ (\mathbf{Z}_2)^\top (\mathbf{G} - \mathbf{Y}_2) + z_3(\mathbf{1}^\top \mathbf{G} - k) + \frac{\rho_1}{2} \|\mathbf{G} - \mathbf{Y}_1\|_2^2 + \frac{\rho_2}{2} \|\mathbf{G} - \mathbf{Y}_2\|_2^2 \\
&+ \frac{\rho_3}{2} (\mathbf{1}^\top \mathbf{G} - k)^2 + h_1(\mathbf{Y}_1) + h_2(\mathbf{Y}_2), \qquad (9)
\end{aligned}
$$

where $\mathbf{Z}_1 \in \mathbb{R}^N, \mathbf{Z}_2 \in \mathbb{R}^N, z_3 \in \mathbb{R}$ are dual variables and (ρ_1, ρ_2, ρ_3) are positive penalty parameters. Function $h_1(\mathbf{Y}_1) = \mathbb{I}_{\{\mathbf{Y}_1 \in \mathcal{S}_b\}}$ and $h_2(\mathbf{Y}_2) = \mathbb{I}_{\{\mathbf{Y}_2 \in \mathcal{S}_p\}}$ are indicator functions, *i.e.*, $\mathbb{I}_{\{a\}} = 0$ when a is true. Otherwise, $\mathbb{I}_{\{a\}} = +\infty$. Following the conventional procedure of the ADMM algorithm, we iteratively update the primal and dual variables, as detailed below.

Step 2.1: Update $\mathbf{Y}_1$. $\mathbf{Y}_1$ is updated via the following minimization problem,

$$
\mathbf{Y}_1 = \underset{\mathbf{Y}_1 \in \mathcal{S}_b}{\arg\min} \frac{\rho_1}{2} \|\mathbf{G} - \mathbf{Y}_1\|_2^2 + (\mathbf{Z}_1)^\top (\mathbf{G} - \mathbf{Y}_1). \qquad (10)
$$

Its solution is obtained by projecting the unconstrained solution of $\mathbf{Y}_1$ to $\mathcal{S}_b$ as

$$
\mathbf{Y}_1 = \mathcal{P}_{\mathcal{S}_b}\big(\mathbf{G} + \frac{1}{\rho_1} \mathbf{Z}_1\big), \qquad (11)
$$

where $\mathcal{P}_{\mathcal{S}_b}(\mathbf{a}) = \min(\mathbf{1}, \max(\mathbf{0}, \mathbf{a}))$ with $\mathbf{a} \in \mathbb{R}^n$ indicates the projection onto the box constraint $\mathcal{S}_b$. Since the objective function of (10) is convex, and the constraint space $\mathcal{S}_b$ is also convex, it is easy to prove that the solution (11) is the optimal solution to Problem (10).

Step 2.2: Update $\mathbf{Y}_2$. $\mathbf{Y}_2$ is updated by the following minimization problem,

$$
\mathbf{Y}_2 = \underset{\mathbf{Y}_2 \in \mathcal{S}_p}{\arg\min} \frac{\rho_2}{2} \|\mathbf{G} - \mathbf{Y}_2\|_2^2 + (\mathbf{Z}_2)^\top (\mathbf{G} - \mathbf{Y}_2). \qquad (12)
$$

According to [36], $\mathbf{Y}_2$ is calculated by

$$
\mathbf{Y}_2 = \mathcal{P}_{\mathcal{S}_p}\big(\mathbf{G} + \frac{1}{\rho_2} \mathbf{Z}_2\big), \qquad (13)
$$

where $\mathcal{P}_{\mathcal{S}_p}(\mathbf{a}) = \frac{\sqrt{n}}{2} \frac{\bar{\mathbf{a}}}{\|\bar{\mathbf{a}}\|} + \frac{1}{2}\mathbf{1}$ with $\bar{\mathbf{a}} = \mathbf{a} - \frac{1}{2}\mathbf{1}$ and $\mathbf{a} \in \mathbb{R}^n$ indicates the projection onto the ℓ_2-sphere constraint $\mathcal{S}_p$. It has been proven in [36] that the solution (13) is the optimal solution to Problem (12).

Step 2.3: Update $\mathbf{G}$. It is infeasible to get a closed-form solution for $\mathbf{G}$, due to the nonlinear function f in $\mathcal{L}(f(\mathbf{x} + \boldsymbol{\delta} \odot \mathbf{G}), y_t)$. We thus update $\mathbf{G}$ by the gradient descent rule,

$$
\begin{aligned}
&\mathbf{G} \leftarrow \mathbf{G} - \eta_{\mathbf{G}} \cdot \frac{\partial L}{\partial \mathbf{G}}, \text{ where } \frac{\partial L}{\partial \mathbf{G}} = 2(\boldsymbol{\delta} \odot \boldsymbol{\delta} \odot \mathbf{G}) + \lambda_1 \frac{\partial \mathcal{L}(f(\mathbf{x} + \boldsymbol{\delta} \odot \mathbf{G}), y_t)}{\partial \mathbf{G}} \\
&+ \rho_1(\mathbf{G} - \mathbf{Y}_1) + \rho_2(\mathbf{G} - \mathbf{Y}_2) + (z_3 + \rho_3(\mathbf{1}^\top \mathbf{G} - k)) \cdot \mathbf{1} + \mathbf{Z}_1 + \mathbf{Z}_2, \qquad (14)
\end{aligned}
$$

where $\eta_{\mathbf{G}} > 0$ and the number of gradient steps will be specified in experiments.

Step 2.4: Update $(\mathbf{Z}_1, \mathbf{Z}_2, z_3)$. The dual variables are updated as follows,

$$\mathbf{Z}_1 \leftarrow \mathbf{Z}_1 + \rho_1(\mathbf{G} - \mathbf{Z}_1),\ \mathbf{Z}_2 \leftarrow \mathbf{Z}_2 + \rho_2(\mathbf{G} - \mathbf{Z}_2),\ z_3 \leftarrow z_3 + \rho_3(\mathbf{1}^\top \mathbf{G} - k). \tag{15}$$

Remark. Since the updates w.r.t $\boldsymbol{\delta}$ (*i.e.*, Step 1) and $\mathbf{G}$ (*i.e.*, Step 2.3) are inexactly solved by gradient descent, the theoretical convergence of Algorithm 1 cannot be guaranteed. However, similar to the inexact ADMM algorithm, we find that Algorithm 1 always converges in our experiments. In terms of the computational complexity, the main costs are computing $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\delta}}$ (see Eq. (7)) and $\frac{\partial \mathcal{L}}{\partial \mathbf{G}}$ (see Eq. (14)), of which the exact costs depend on the attacked model f.

3.4 Two Extensions of SAPF

The factorization of $\boldsymbol{\epsilon}$ in Eq. (2) provides the extra flexibility to impose different constraints on perturbation magnitudes or selection factors (*i.e.*, perturbed positions), so as to achieve some desired performance of the attacker. Here we provide two case studies, including *group-wise sparsity* and *visual imperceptibility*.

Group-wise Sparsity. Model (3) is a natural combination of the pixel-wise adversarial attack and sparsity. In the literature of sparsity, the group-wise sparsity [40,3] is well studied and shows promising performance on encouraging to select grouped variables. One recent work called *StrAttack* [38] introduced the group-wise sparsity into the adversarial attack, providing some insights about the influences of different regions of one image on the adversarial attack. Without loss of generality, we assume the input image $\mathbf{x}$ is split into m sub-regions $\{\mathbf{x}^i\}_{i=1}^m$. $\boldsymbol{\delta}^i$ and $\mathbf{G}^i$ denote perturbation magnitudes and selection factors corresponding to the i-th region $\mathbf{x}^i$, respectively. Through the factorization of magnitude and selection factors, the group-wise sparsity can be realized by minimizing $\sum_{i=1}^m \|\mathbf{G}^i\|_2$, which is added onto model (3), leading to

$$\min_{\boldsymbol{\delta}, \mathbf{G} \in \{0,1\}^N} \|\boldsymbol{\delta} \odot \mathbf{G}\|_2^2 + \lambda_1 \mathcal{L}(f(\mathbf{x} + \boldsymbol{\delta} \odot \mathbf{G}), y_t) + \lambda_2 \sum_{i=1}^m \|\mathbf{G}^i\|_2,\ \text{s.t. } \mathbf{1}^\top \mathbf{G} = k, \tag{16}$$

where $\lambda_2 > 0$ controls the significance of the group-wise sparsity. This problem can be solved using Algorithm 1, by modifying the update of $\mathbf{G}$ (see Eq. (14)) by adding the gradient of $\lambda_2 \sum_{i=1}^m \|\mathbf{G}^i\|_2$ w.r.t. $\mathbf{G}$. It is denoted as **SAPF-GS**.

Visual Imperceptibility. The visual imperceptibility is important for practical adversarial learning. As shown in [19,26], the sensitiveness of humans to different image regions varies according to pixel values. Thus, it is useful to assign relative smaller perturbations to regions with the higher sensitiveness. This strategy can be naturally incorporated into the proposed model (3), as follows

$$\min_{\boldsymbol{\delta}, \mathbf{G}} \|\boldsymbol{w} \odot \boldsymbol{\delta} \odot \mathbf{G}\|_2^2 + \lambda_1 \mathcal{L}(f(\mathbf{x} + \boldsymbol{\delta} \odot \mathbf{G}), y_t),\ \text{s.t. } \mathbf{1}^\top \mathbf{G} = k,\ \mathbf{G} \in \{0,1\}^N, \tag{17}$$

where $\boldsymbol{w} \in [0, 1]^N$ denotes the pre-defined weight at each pixel. The minimization of $\|\boldsymbol{w} \odot \boldsymbol{\delta} \odot \mathbf{G}\|_2^2$ encourages to assign relative smaller perturbations at pixels with higher weights, and vice-versa. The derivation of $\boldsymbol{w}$ will be discussed in experiments. Problem (17) can be directly solved using Algorithm 1 by slightly modifying the gradients w.r.t. $\boldsymbol{\delta}$ and $\mathbf{G}$. Problem (17) is denoted as **SAPF-VI**.

4 Experiments

In this section, we conduct experiments on CIFAR-10 [18] and ImageNet [11]. We compare our method with several state-of-the-art sparse adversarial attack algorithms, including five pixel-wise attack algorithms (C&W-ℓ_0 [7], One-Pixel-Attack [30], SparseFool [27], CornerSearch [10] and PGD ℓ_0+ℓ_∞ [10]), and one group-wise attack algorithm (StrAttack [38]).

4.1 Experimental Settings

Database and Classification Model. CIFAR-10 has 50k training images and 10k validation images, covering 10 classes. Following [38], we randomly select 1,000 images from the validation set as input. Each image has 9 target classes except its ground-truth class. Thus a total number of 9,000 adversarial examples need to be learned for each adversarial attack method. ImageNet contains 1,000 classes, with 1.28 million images for training and 50k images for validation. We randomly choose 100 images covering 100 different classes from the validation set. To reduce the time complexity, we randomly select 9 target classes for each image in ImageNet, resulting in 900 adversarial examples. For the classification model f, on CIFAR-10, we follow the setting of C&W [7] and train a network that consists of four convolution layers, three fully-connected layers, and two max-pooling layers. The input size of the network is $32 \times 32 \times 3$. It achieves 79.51% top-1 classification accuracy on the validation set. On ImageNet, we use a pre-trained Inception-v3 network[3] [32] with 77.45% top-1 classification accuracy. The input size of the network is $299 \times 299 \times 3$.

Parameter Settings. In the proposed model (3), the trade-off hyper-parameter λ_1 can effect the perturbation magnitude and the attack success rate. Following C&W [7], we use a modified binary search to find an appropriate λ_1. Specifically, λ_1 is initialized as 0.001 on CIFAR-10 and 0.01 on ImageNet, respectively. The lower and upper bound of λ_1 are set to 0 and 100, respectively. The binary search of λ_1 stops when generating a successful attack or exceeding a maximum search times (*e.g.*, 6 times in our experiments). Besides, in Algorithm 1, the maximum number of iterations is set to 10. During each iteration, both $\mathbf{G}$ and $\boldsymbol{\delta}$ are updated by the gradient-descent method (see Eqs. (7) and (14)) for 2000 steps with an initial step size 0.1, and the step size decays for every 50 steps with the decay rate of 0.9. Besides, as shown in Algorithm 1, we adopt the simple initialization that $\mathbf{G} = \mathbf{1}$ and $\boldsymbol{\delta} = \mathbf{0}$. It ensures the fair chance for each individual pixel to be

[3] Downloaded from https://download.pytorch.org/models/inception_v3_google-1a9a5a14.pth

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	One-Pixel [30]	15	3	1.572	0.956	0.676	5.489	3	2.191	1.286	0.817	0.7	3	2.662	1.539	0.922
	CornerSearch [10]	60.4	537	69.704	3.335	0.336	59.3	549	73.64	3.481	0.342	63.2	561	77.570	3.621	0.346
	PGD $\ell_0 + \ell_\infty$ [10]	99.4	555	18.112	0.966	0.116	98.6	555	23.172	1.169	0.123	99.3	555	26.815	1.349	0.125
	SparseFool [27]	100	**255**	11.873	0.665	0.047	99.9	553	25.81	1.041	**0.047**	99.8	852	39.674	1.339	**0.047**
	C&W-ℓ_0 [7]	100	614	6.948	0.428	0.086	100	603	13.071	0.805	0.157	100	598	18.603	1.141	0.221
	StrAttack [38]	100	391	4.936	0.296	0.053	100	543	9.494	0.524	0.087	100	476	12.436	0.771	0.137
	SAPF (Ours)	**100**	387	**4.612**	**0.251**	**0.039**	**100**	**539**	**8.513**	**0.435**	0.064	**100**	**471**	**10.392**	**0.604**	0.095
ImageNet	One-Pixel [30]	0	3	1.192	0.804	0.664	0	3	1.881	1.179	0.833	0	3	2.562	1.509	0.933
	CornerSearch [10]	4	58658	5962.457	28.06	0.436	1.3	58792	6018.307	28.29	0.435	2	58920	6076.068	28.53	0.437
	PGD $\ell_0 + \ell_\infty$ [10]	95	56922	798.888	4.205	0.063	95.6	56919	854.674	4.508	0.063	96	**56920**	925.272	4.901	0.063
	SparseFool [27]	97	**34205**	174.146	0.918	**0.005**	80.6	59940	305.182	1.219	0.005	46	82576	420.440	1.450	0.005
	C&W-ℓ_0 [7]	100	73407	133.790	0.786	0.051	100	70885	199.203	1.117	0.058	100	69947	269.097	1.463	0.065
	StrAttack [38]	100	38354	77.279	0.694	0.062	100	58581	127.585	0.974	0.081	100	67348	171.248	1.276	0.100
	SAPF (Ours)	**100**	37275	**70.253**	**0.586**	0.038	**100**	**56218**	**112.155**	**0.719**	**0.037**	**100**	65250	**150.552**	**0.872**	**0.041**

Table 1: Results of targeted sparse adversarial attack on CIFAR-10 and ImageNet, evaluated by ASR and ℓ_p-norm ($p = 0, 1, 2, \infty$) of the learned perturbation. The best ℓ_p-norm among methods that achieve more than 90% ASR is shown in bold.

perturbed, and avoids the uncertainty due to the random initialization, such that our reported experimental results can be easily reproduced. Hyper-parameters (ρ_1, ρ_2, ρ_3) in ADMM (see Eq. (15)) are initialized as $(5 \times 10^{-3}, 5 \times 10^{-3}, 10^{-4})$ on both CIFAR-10 and ImageNet, and increase by $\rho_i \leftarrow 1.01 \times \rho_i, i = 1, 2, 3$ after each iteration. The maximum values of (ρ_1, ρ_2, ρ_3) are set to $(20, 10, 100)$ on both databases. The number of perturbed positions k is a key hyper-parameter for sparse attack. It can be explicitly controlled through the cardinality constraint in our SAPF method (see model (3)), while not the case in most existing sparse attack methods. To ensure the fair comparison, in experiments we firstly run the baseline C&W-ℓ_0 with 100% ASR. The ℓ_0-norm of C&W-ℓ_0 under the *average* case (see the next paragraph) serves as the reference number to tune the k values for other methods (including SAPF). The similar level of sparsity of all compared methods facilitates the comparison using ℓ_2 and ℓ_∞-norm.

Evaluation Metrics. The ℓ_p-norm ($p = 0, 1, 2, \infty$) of perturbations and the attack success rate (ASR) are used to evaluate the attack performance of different methods. In our experiments, we keep increasing the upper bound of ℓ_p-norm of perturbations until the attack is success. In other words, we compare different attack algorithms in terms of the ℓ_p-norm of perturbations under 100% ASR[4]. Moreover, for each image, similar to C&W [7], we evaluate three different cases, *i.e.*, **average case**: the average performance of all 9 target classes; **best case**: the performance w.r.t. the target class that is the easiest to attack; and **worst case:** the performance w.r.t. the target class that is the most difficult to attack.

4.2 Experimental Comparisons between SAPF and Other Methods

Results on CIFAR-10. The average ℓ_p-norm and the ASR of the learned perturbation on CIFAR-10 under three different cases are given in Table 1. From the table, we see that our method achieve 100% attack success rate under all

[4] Note that some sparse attack methods fail to generate 100% ASR in our experiments.

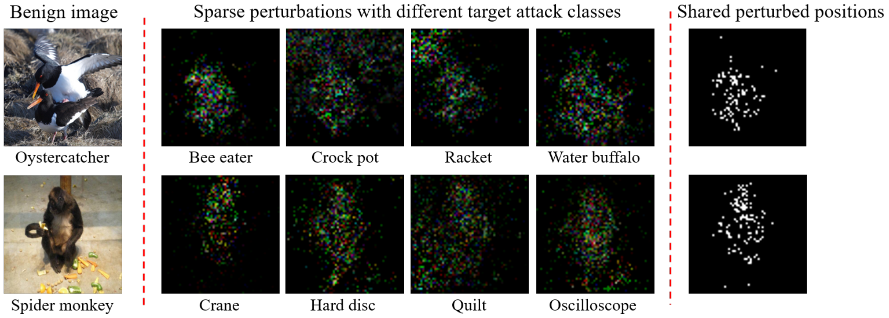

Fig. 2: Examples of perturbations generated by the proposed SAPF method. (**Left-most column**): Two benign images from ImageNet with their ground-truth class labels given below; (**2nd - 5th columns**): the generated sparse adversarial perturbations with different target attack classes given below; (**Right-most column**): the common perturbed pixels of four targeted adversarial attacks.

three cases. The ℓ_∞-norm of the One-Pixel-Attack is the largest among all algorithms and it achieves the lowest attack success rate, *e.g.*, it only achieves 15% ASR under the best case. Thus, it is hard to perform targeted adversarial attacks by only perturbing one pixel (the $\ell_0 = 3$ relates to three channels), even on the tiny database CIFAR-10. The CornerSearch also fails to generate 100% success attack rate. Comparing to all adversarial attack algorithms except One-Pixel-Attack, our method achieves the best ℓ_1-norm and ℓ_2-norm under all three cases. This demonstrates the effectiveness of the proposed method. The SparseFool achieves lower ℓ_∞-norm under the average and worst cases. However, its ℓ_0-norm, ℓ_1-norm and ℓ_2-norm are significantly higher than our method. The C&W-ℓ_0, StrAttack and our method all achieve 100% attack success rate. However, by factorizing the perturbation into positions and magnitude and jointly optimizing them, our model significantly outperforms the C&W-ℓ_0 and StrAttack, and achieves 100% ASR with the lowest ℓ_0-norm, ℓ_1-norm, ℓ_2-norm and ℓ_∞-norm. These demonstrate the superiority of our proposed method.

Results on ImageNet. The numerical results of different adversarial attack algorithms on ImageNet are given in Table 1. Seen from it, our method achieves 100% attack success rate under all three cases. The One-Pixel-Attack and CornerSearch algorithms fail to perform targeted adversarial attack on ImageNet under all three cases. The SparseFool method also fails to generate successful attack for many images, especially for the worst case where its ASR is only 46%. The C&W-ℓ_0 and StrAttack algorithms also achieve 100% ASR under all three cases. However, our method achieves the same 100% ASR with the least number of perturbed positions. And the ℓ_1-norm, ℓ_2-norm and ℓ_∞-norm of C&W-ℓ_0 and StrAttack are significant higher than our method. The SparseFool obtains the

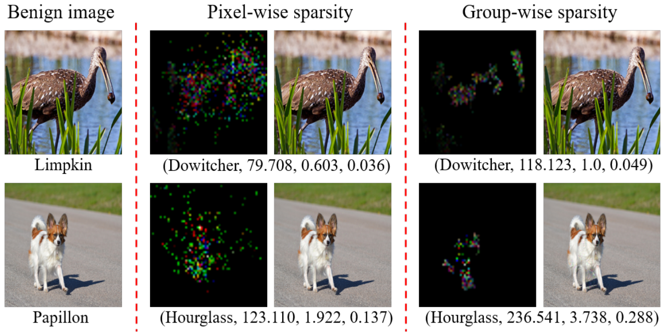
Benign image
Pixel-wise sparsity
Group-wise sparsity
Limpkin
(Dowitcher, 79.708, 0.603, 0.036)
(Dowitcher, 118.123, 1.0, 0.049)
Papillon
(Hourglass, 123.110, 1.922, 0.137)
(Hourglass, 236.541, 3.738, 0.288)

Fig. 3: Examples of perturbations with group-wise sparsity. (**Left column**): benign images with their ground-truth classes given below. (**Middle** and **Right column**) show the learned perturbations by SAPF-GS (*i.e.*, model (16)) with $\lambda_2 = 0$ and $\lambda_2 = 10$, respectively. The text under each perturbation indicates its target attack class and $\ell_1, \ell_2, \ell_\infty$-norm, respectively.

lowest ℓ_∞-norm. However, it fails to generate 100% ASR, and its ℓ_1-norm and ℓ_2-norm are significantly higher than ours.

Visualization of the Learned Sparse Perturbations. The sparse perturbed positions are adaptively determined during the optimization process. For the better understanding of the learned perturbed positions, we present two examples of visualizing the learned adversarial perturbations in Fig. 2. The benign images in the first column of Fig. 2 can be correctly classified by the Inception-v3 model. However, by adding small and sparse adversarial perturbations (*i.e.*, the images from column 2 to column 5) onto benign images, the corresponding adversarial images successfully fool the Inception-v3 model. One interesting phenomenon is that the positions of the learned perturbation are highly related to the discriminative image regions. For example, in the second row, when performing a targeted attack on the benign image "Spider monkey" with target attack class "Crane", the learned sparse perturbation mainly locates in the image regions related to the object areas. A similar phenomenon can also be observed in other images and different target attack classes. For each row, the right-most plot highlights the positions that are always perturbed under four different target attacks (*i.e.*, 2nd - 5th column). We observe that when attacking certain benign image to different target classes, the learned sparse perturbations w.r.t. different target classes share common perturbed positions.

Degree of Sparsity. Here we evaluate the impact of different degrees of sparsity in sparse attack. Specifically, we evaluate SAPF (*i.e.*, model (3)) with different values of the cardinality k. For each k, we keep increasing the magnitude of the perturbation until the attack is success. Fig. 4 shows ℓ_p-norms of the perturbation generated by our SAPF attack w.r.t. the number of perturbed positions k. As k increases, more positions are perturbed, and the magnitudes of most perturbations decrease, leading to the decreasing of ℓ_∞-norm. Besides, since the

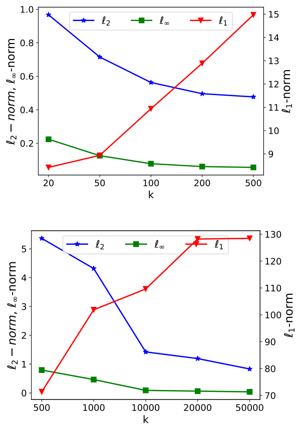

Fig. 4: $\ell_1, \ell_2, \ell_\infty$-norms of perturbations generated by SAPF w.r.t. the number of perturbed positions k on CIFAR-10 (**top**) and ImageNet (**bottom**).

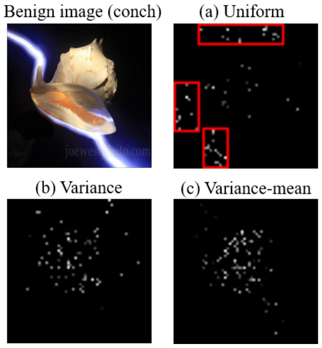

Fig. 5: Perturbation visualization of different weighting strategies in SAPF-VI (*i.e.*, model (17)). The top-left image is the benign image with class "conch". The other three images are the learned perturbed positions with three weighting strategies. The target attack class is "fountain".

magnitude of each single perturbation is much smaller than 1, ℓ_2-norm also decreases; in contrast, ℓ_1-norm increases, as the increasing from new perturbations is larger than the decreasing of old perturbations.

4.3 Results of Group-wise Sparsity and Visual Imperceptibility

SAPF with Group-wise Sparsity. Fig. 3 visualizes the learned perturbations without and with group-wise sparsity in the middle and right columns, respectively. The number of perturbed positions (*i.e.*, k) is the same for these two cases. For group-wise sparsity, we split input image into 350 sub-regions via super-pixel segmentation [1]. Comparing the learned perturbations in the middle and right columns, the learned perturbations with group-wise constraint (*i.e.*, the plots in the right column) are more concentrated and gather around discriminative object regions. It helps to explore regions that are sensitive to adversarial attacks. Meanwhile, the perturbations with the group-wise sparsity get larger (see the ℓ_p-norm under each perturbation in Fig. 3), but still imperceptible, due to the trade-off between the group-wise sparsity and the other two terms in the objective function (16). This trade-off can be flexibly adjusted through the hyper-parameter λ_2 in (16) in practice.

SAPF with Visual Imperceptibility. For visual imperceptibility, we present the perturbed positions learned by model (17) with different weighting strategies in Fig. 5. We consider three weighting strategies: "uniform" that assigns equal weight to each position; "variance" weight $\boldsymbol{w}_i = 1/var(\mathbf{x}_i)$ [26], where $var(\mathbf{x}_i) = \sqrt{\sum_{\mathbf{x}_j \in \mathcal{S}_i} (\mathbf{x}_j - \mu_i)^2 / n^2}$ and $\mathcal{S}_i$ denotes the $n \times n$ neighborhoods of position i,

$\mu_i = \sum_{\mathbf{x}_j \in \mathcal{S}_i} \mathbf{x}_j / n^2$; and "variance-mean" weight $\boldsymbol{w}_i = 1/\left(var(\mathbf{x}_i) \times \mu_i\right)$, n is set to 3 empirically. For better visualization, we highlight the top-1% perturbed positions with the largest magnitudes. The "uniform" weight treats each position equally, and its learned perturbed positions may be located at uniform background regions where humans are more sensitive (*e.g.*, the red box areas in Fig. 5 (b)). Considering that humans are more sensitive to perturbation at regions with lower variance, the "variance" weight encourages the model to assign less perturbation to positions with lower variance. And its learned perturbed positions mainly focus on the object regions with higher variance. The "variance-mean" weight further considers the brightness around each position, and its learned perturbed positions are barely located at the uniform background regions with lower variance and lower brightness. Thus, the pixel weight $\boldsymbol{w}$ in model (17) influences the learned perturbed positions, and it is interesting to explore different weighting strategies to further enhance visual imperceptibility.

4.4 Supplementary Materials

Due to the space limit, some important contents have to be presented in supplementary materials, including: **1)** the results of attacking the adversarially trained model on CIFAR-10; **2)** the running time of different attack methods; **3)** detailed discussions of three important problems, including the values of sparse adversarial attack, the most important contribution of this work, other extensions of the proposed method.

5 Conclusion

This work provided a new perspective of the adversarial perturbation that each perturbation could be factorized by two characteristics, *i.e.*, magnitude and position. This new perspective enables to formulate the the sparse adversarial attack as a mixed integer programming (MIP) problem, which jointly optimizes perturbation magnitudes and perturbed positions. The degree of sparsity is explicitly controlled via the cardinality constraint on the perturbed positions. We developed an efficient and effective optimization algorithm by equivalently reformulating the MIP problem as a continuous optimization problem. Experimental evaluations on two benchmark databases demonstrate the superiority of the proposed method to state-of-the-art sparse adversarial attack methods. Besides, we visualized that the learned sparse positions closely related to the discriminative regions, and also showed that the proposed model is flexible to incorporate different constraints on perturbation magnitudes or perturbed positions, such as group-wise sparsity and visual imperceptibility.

Acknowledgement This work is supported by Tencent AI Lab. The participation of Yujiu Yang is supported by The Key Program of National Natural Science Foundation of China under Grant No. U1903213.

References

1. Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S.: Slic superpixels compared to state-of-the-art superpixel methods. IEEE transactions on pattern analysis and machine intelligence **34**(11), 2274–2282 (2012)
2. Akhtar, N., Mian, A.: Threat of adversarial attacks on deep learning in computer vision: A survey. IEEE Access **6**, 14410–14430 (2018)
3. Bach, F., Jenatton, R., Mairal, J., Obozinski, G., et al.: Optimization with sparsity-inducing penalties. Foundations and Trends® in Machine Learning **4**(1), 1–106 (2012)
4. Bai, J., Chen, B., Li, Y., Wu, D., Guo, W., Xia, S.t., Yang, E.h.: Targeted attack for deep hashing based retrieval. In: ECCV (2020)
5. Bibi, A., Wu, B., Ghanem, B.: Constrained k-means with general pairwise and cardinality constraints. arXiv preprint arXiv:1907.10410 (2019)
6. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends® in Machine learning **3**(1), 1–122 (2011)
7. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 39–57. IEEE (2017)
8. Chen, H., Zhang, H., Chen, P.Y., Yi, J., Hsieh, C.J.: Show-and-fool: Crafting adversarial examples for neural image captioning. arXiv preprint arXiv:1712.02051 (2017)
9. Chen, W., Zhang, Z., Hu, X., Wu, B.: Boosting decision-based black-box adversarial attacks with random sign flip. In: Proceedings of the European Conference on Computer Vision (2020)
10. Croce, F., Hein, M.: Sparse and imperceivable adversarial attacks. In: ICCV. pp. 4724–4732 (2019)
11. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR. pp. 248–255. Ieee (2009)
12. Dong, Y., Su, H., Wu, B., Li, Z., Liu, W., Zhang, T., Zhu, J.: Efficient decision-based black-box adversarial attacks on face recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7714–7722 (2019)
13. Feng, Y., Chen, B., Dai, T., Xia, S.: Adversarial attack on deep product quantization network for image retrieval. In: AAAI (2020)
14. Feng, Y., Wu, B., Fan, Y., Li, Z., Xia, S.: Efficient black-box adversarial attack guided by the distribution of adversarial perturbations. arXiv preprint arXiv:2006.08538 (2020)
15. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. ICLR (2015)
16. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
17. Karmon, D., Zoran, D., Goldberg, Y.: Lavan: Localized and visible adversarial noise. ICML (2018)
18. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Tech. rep., Citeseer (2009)
19. Legge, G.E., Foley, J.M.: Contrast masking in human vision. Josa **70**(12), 1458–1471 (1980)
20. Li, T., Wu, B., Yang, Y., Fan, Y., Zhang, Y., Liu, W.: Compressing convolutional neural networks via factorized convolutional filters. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3977–3986 (2019)

21. Li, Y., Wu, B., Feng, Y., Fan, Y., Jiang, Y., Li, Z., Xia, S.: Toward adversarial robustness via semi-supervised robust training. arXiv preprint arXiv:2003.06974 (2020)
22. Li, Y., Yang, X., Wu, B., Lyu, S.: Hiding faces in plain sight: Disrupting ai face synthesis with adversarial perturbations. arXiv preprint arXiv:1906.09288 (2019)
23. Liu, A., Huang, T., Liu, X., Xu, Y., Ma, Y., Chen, X., Maybank, S., Tao, D.: Spatiotemporal attacks for embodied agents. In: European Conference on Computer Vision (2020)
24. Liu, A., Liu, X., Fan, J., Ma, Y., Zhang, A., Xie, H., Tao, D.: Perceptual-sensitive gan for generating adversarial patches. In: 33rd AAAI Conference on Artificial Intelligence (2019)
25. Liu, A., Wang, J., Liu, X., Cao, b., Zhang, C., Yu, H.: Bias-based universal adversarial patch attack for automatic check-out. In: European Conference on Computer Vision (2020)
26. Luo, B., Liu, Y., Wei, L., Xu, Q.: Towards imperceptible and robust adversarial example attacks against neural networks. In: AAAI (2018)
27. Modas, A., Moosavi-Dezfooli, S.M., Frossard, P.: Sparsefool: a few pixels make a big difference. In: CVPR. pp. 9087–9096 (2019)
28. Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A.: The limitations of deep learning in adversarial settings. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 372–387. IEEE (2016)
29. Sarikaya, R., Hinton, G.E., Deoras, A.: Application of deep belief networks for natural language understanding. IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP) **22**(4), 778–784 (2014)
30. Su, J., Vargas, D.V., Sakurai, K.: One pixel attack for fooling deep neural networks. IEEE Transactions on Evolutionary Computation (2019)
31. Sun, Y., Liang, D., Wang, X., Tang, X.: Deepid3: Face recognition with very deep neural networks. arXiv preprint arXiv:1502.00873 (2015)
32. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: CVPR. pp. 2818–2826 (2016)
33. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. ICLR (2014)
34. Wang, H., Wang, Y., Zhou, Z., Ji, X., Gong, D., Zhou, J., Li, Z., Liu, W.: Cosface: Large margin cosine loss for deep face recognition. In: CVPR. pp. 5265–5274 (2018)
35. Wu, B., Chen, W., Fan, Y., Zhang, Y., Hou, J., Liu, J., Zhang, T.: Tencent ml-images: A large-scale multi-label image database for visual representation learning. IEEE Access **7**, 172683–172693 (2019)
36. Wu, B., Ghanem, B.: lp-box admm: A versatile framework for integer programming. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) (2018)
37. Wu, B., Shen, L., Zhang, T., Ghanem, B.: Map inference via l2-sphere linear program reformulation. International Journal of Computer Vision pp. 1–24 (2020)
38. Xu, K., Liu, S., Zhao, P., Chen, P.Y., Zhang, H., Erdogmus, D., Wang, Y., Lin, X.: Structured adversarial attack: Towards general implementation and better interpretability. ICLR (2019)
39. Xu, Y., Wu, B., Shen, F., Fan, Y., Zhang, Y., Shen, H.T., Liu, W.: Exact adversarial attack to image captioning via structured output learning with latent variables. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4135–4144 (2019)
40. Yuan, M., Lin, Y.: Model selection and estimation in regression with grouped variables. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **68**(1), 49–67 (2006)

41. Zhao, P., Liu, S., Wang, Y., Lin, X.: An admm-based universal framework for adversarial attacks on deep neural networks. In: 2018 ACMMM. pp. 1065–1073. ACM (2018)