PRACTICAL ORDER ATTACK IN DEEP RANKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent studies have unveiled the vulnerabilities of deep ranking models, where an imperceptible perturbation could trigger dramatic changes in the ranking result. However, previous attempts focus on manipulating *absolute ranks* of certain candidates, while the possibility of adjusting their *relative order* remains underexplored. The objective of this paper is to formalize and practically implement a new adversarial attack against deep ranking systems, *i.e.*, the *Order Attack*, which covertly alters the *relative order* of a selected set of candidates according to a permutation vector predefined by the attacker, with only limited interference to other unrelated candidates. Although this *Order Attack* can be formulated as a triplet-style loss constraint imposing an inequality chain that reflects the attacker's desired permutation, direct optimization of such loss is inapplicable in a real-world black-box attack scenario due to the inaccessibility of gradients, limited query budget, truncated ranking results, and lack of similarity scores. To address these challenges, we propose a new *Short-range Ranking Correlation* metric as a surrogate objective function to approximate Kendall's ranking correlation while maintaining robustness to these practical limitations. The proposed white-box and black-box attacks are evaluated on the Fashion-MNIST and Stanford-Online-Products datasets. Moreover, the black-box attack is successfully implemented on a major e-commerce platform. Extensive quantitative and qualitative experimental evaluations demonstrate the effectiveness of our proposed methods, revealing deep ranking systems' vulnerability to the *Order Attack*.

1 INTRODUCTION

Thanks to the widespread applications of deep neural networks (Krizhevsky et al., 2012; He et al., 2016) in the learning-to-rank tasks (Wang et al., 2014; Schroff et al., 2015), deep ranking algorithms have witnessed significant progress, but unfortunately they have also inherited the long-standing adversarial vulnerabilities of neural networks. Consider the "search by image" application for example, an imperceptible adversarial perturbation to the query image is often sufficient to intentionally alter the ranking results of candidate images. Typically, such adversarial examples can be designed to cause the ranking model to "misrank" (Liu et al., 2019; Li et al., 2019a) (*i.e.*rank items incorrectly), or purposefully raise or lower the ranks of selected candidates (Zhou et al., 2020).

Since "misranking" can be interpreted as deliberately lowering the ranks of well-matching candidates, previous attacks on ranking models unanimously focus on changing the *absolute ranks* of a set of candidates, while neglecting the manipulation of *relative order* among them. However, the *relative order* can be critical in some applications, such as content-based image retrieval (Smeulders et al., 2000) on e-commerce platforms, where potential customers attempt to find the exactly matching merchandise via the search-by-image functionality. As shown in Fig. 1, an attacker may want to manipulate the sales of products A, B, C, D, and E by changing the *relative order* of A to E into A $\prec$ E $\prec$ D $\prec$ C $\prec$ B in the search-by-image query result, which can be achieved by adding an imperceptible adversarial perturbation to the query image. This attack does not aim to incur "misranking" or significantly change the *absolute ranks* of the chosen candidates, instead it intentionally attempts to subtly change the *relative order* of them without introducing conspicuous abnormality. Challenges posed by such attacks may stimulate the creation of more robust ranking models.

Specifically, we propose the *Order Attack* (OA), a new adversarial attack problem in deep ranking. Given a query image $\boldsymbol{q} \in [0,1]^D$, a set of selected candidates $\mathbb{C} = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k\}$, and a predefined permutation vector $\mathbf{p} = [p_1, p_2, \ldots, p_k]$, *Order Attack* aims to find an imperceptible perturbation $\boldsymbol{r}$

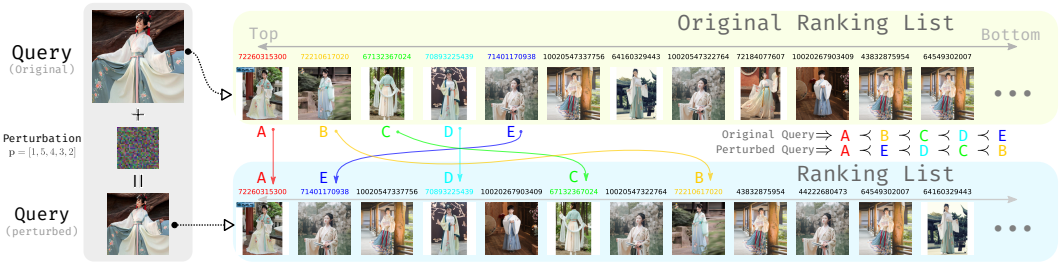

Figure 1: Showcase of a practical *Order Attack* (OA) against "JD SnapShop", a major online retailing e-commerce platform. Numbers atop candidate images are Stock Keep Unit (SKU) IDs.

($\|\boldsymbol{r}\|_\infty \leqslant \varepsilon$ and $\tilde{\boldsymbol{q}} = \boldsymbol{q} + \boldsymbol{r} \in [0,1]^D$), so that $\tilde{\boldsymbol{q}}$ as the adversarial query can convert the *relative order* of the selected candidates into $\boldsymbol{c}_{p_1} \prec \boldsymbol{c}_{p_2} \prec \cdots \prec \boldsymbol{c}_{p_k}$. For example, a successful OA with $\mathbf{p} = [1, 5, 4, 3, 2]$ will result in $\boldsymbol{c}_1 \prec \boldsymbol{c}_5 \prec \boldsymbol{c}_4 \prec \boldsymbol{c}_3 \prec \boldsymbol{c}_2$, as shown in Fig. 1.

To implement OA, we first (unrealistically) assume the *white-box* threat model (*i.e.*, the ranking model details, *incl.* the gradient, are accessible to the attacker) and present a triplet-style loss function. Conventionally, a typical deep ranking model (Wang et al., 2014; Zhou et al., 2019; Schroff et al., 2015; Kim et al., 2019) maps the query and candidates onto a common embedding space, and determines the ranking list according to the pairwise similarity between the query and these candidates. Based on these, OA can be formulated as the optimization of a triplet-style loss function, which simultaneously adjusts the similarity scores between the query and the selected candidates according to the inequalities representing the desired *relative order*. Additionally, a semantics-preserving penalty term (Zhou et al., 2020) is also included to limit conspicuous changes in ranking results. Subsequently, the overall loss function can be optimized with gradient methods such as PGD (Madry et al., 2017) to find the desired adversarial example.

However, in a real-world *black-box* attack scenario, practical limitations invalidate the previous white-box method, due to (1) *Gradient inaccessibility*. The gradient of the loss *w.r.t* the input is inaccessible, as the network architecture and parameters are unknown; (2) *Limited query budget*. Repeated, intensive queries within a short time frame may be identified as threats, *e.g.*, Denial of Service (DoS) attack. Therefore, it is preferable to construct adversarial examples within a reasonable amount of queries; (3) *Truncated ranking results*. In practice, a ranking system only presents the top-N ranking results to the clients; (4) *Lack of similarity (or distance) scores*. Exact similarity scores rarely appear in the truncated ranking results. To overcome these limitations, we propose the "Short-range Ranking Correlation" (SRC) metric to measure the alignment between a desired permutation and the practical ranking result shown to clients, as a practical approximation of Kendall's ranking correlation (Kendall, 1945). Though non-differentiable, SRC can be used as a surrogate objective for black-box OA and optimized by an appropriate black-box optimizer.

To validate the white-box and black-box OA, we conduct comprehensive experiments on Fashion-MNIST and Stanford-Online-Product datasets. To illustrate the viability of the black-box OA in practice, we also qualitatively showcase successful attacks against the "JD SnapShop" (JingDong, 2020), a major retailing e-commerce platform based on content-based image retrieval. Extensive quantitative and qualitative evaluations illustrate the effectiveness of our proposed OA.

To the best of our knowledge, this is the first work that tampers the *relative order* in deep ranking. We believe our contributions include, **(1)** the formulation of *Order Attack* (OA), a new adversarial attack that covertly alters the *relative order* among selected candidates; **(2)** a triplet-style loss for ideal white-box OA, which can be optimized by PGD; **(3)** a *Short-range Ranking Correlation* (SRC) metric as a surrogate objective for practical black-box OA; **(4)** extensive evaluations of both OAs including a successful demonstration on a major online retailing e-commerce platform.

2 Adversarial Order Attack

Typically, a deep ranking model is built upon deep metric learning (Wang et al., 2014; Schroff et al., 2015; Zhou et al., 2019; Kim et al., 2019). Given a query $\boldsymbol{q}$ and a set of candidates $\mathbb{C} =$

$\{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_m\}$ selected from database $\mathbb{D}$ ($\mathbb{C} \subset \mathbb{D}$), a deep ranking model f evaluates the distance between every pair of query and candidate, *i.e.*$f : \mathcal{I} \times \mathcal{I} \mapsto \mathbb{R}$ where $\mathcal{I} = [0,1]^D$. Thus, by comparing all the pairwise distances $\{f(\boldsymbol{q}, \boldsymbol{c}_i)|i = 1, 2, \ldots, k\}$, the whole candidate set can be ranked with respect to the given query. For instance, the model outputs the ranking list $\boldsymbol{c}_1 \prec \boldsymbol{c}_2 \prec \cdots \prec \boldsymbol{c}_k$ if it determines $f(\boldsymbol{q}, \boldsymbol{c}_1) < f(\boldsymbol{q}, \boldsymbol{c}_2) < \cdots < f(\boldsymbol{q}, \boldsymbol{c}_k)$.

Based on these, *Order Attack* (OA) aims to find an imperceptible perturbation $\boldsymbol{r}$ ($\|\boldsymbol{r}\|_\infty \leqslant \varepsilon$ and $\tilde{\boldsymbol{q}} = \boldsymbol{q} + \boldsymbol{r} \in \mathcal{I}$), so that $\tilde{\boldsymbol{q}}$ as the adversarial query can convert the *relative order* of the selected candidates into $\boldsymbol{c}_{p_1} \prec \boldsymbol{c}_{p_2} \prec \cdots \prec \boldsymbol{c}_{p_k}$, where $\mathbf{p} = [p_1, p_2, \ldots, p_k]$ is a permutation vector predefined by the attacker. In particular, we assume that the attacker is inclined to select the candidate set $\mathbb{C}$ from the top-N ranked candidates $\mathbb{X}$ ($N \geqslant k$, where N, the length of the truncated ranking list, is called a "visible range"). The white-box and black-box OA will be discussed in Sec.2.1 and Sec.2.2 respectively. For sake of brevity, we let $\Omega_{\boldsymbol{q}} = \{\boldsymbol{r} |\ \boldsymbol{q} + \boldsymbol{r} \in \mathcal{I},\ \|\boldsymbol{r}\|_\infty \leqslant \varepsilon\}$.

2.1 Triplet-style Loss Function for White-Box Order Attack

During the training process, a typical deep ranking model f involves a triplet (anchor $\boldsymbol{q}$, positive $\boldsymbol{c}_\text{p}$, negative $\boldsymbol{c}_\text{n}$) in each iteration, and in order to rank $\boldsymbol{c}_\text{p}$ ahead of $\boldsymbol{c}_\text{n}$, the model is penalized when $f(\boldsymbol{q}, \boldsymbol{c}_\text{p}) + \beta < f(\boldsymbol{q}, \boldsymbol{c}_\text{n})$ does not hold. This inequality can be reformulated exploiting the form of a hinge loss, resulting in the triplet ranking loss function $L_\text{triplet}(\boldsymbol{q}, \boldsymbol{c}_\text{p}, \boldsymbol{c}_\text{n}) = [\gamma + f(\boldsymbol{q}, \boldsymbol{c}_\text{p}) - f(\boldsymbol{q}, \boldsymbol{c}_\text{n})]^+$ where $[\cdot]^+ = \max(0, \cdot)$, and γ denotes the margin, a positive constant hyper-parameter.

Likewise, to implement the OA, we decompose the inequality chain prescribed by the permutation vector $\mathbf{p}$, namely $f(\tilde{\boldsymbol{q}}, \boldsymbol{c}_{p_1}) < f(\tilde{\boldsymbol{q}}, \boldsymbol{c}_{p_2}) < \cdots < f(\tilde{\boldsymbol{q}}, \boldsymbol{c}_{p_k})$ into $\binom{k}{2} = k(k-1)/2$ inequalities, *i.e.*, $f(\tilde{\boldsymbol{q}}, \boldsymbol{c}_{p_i}) < f(\tilde{\boldsymbol{q}}, \boldsymbol{c}_{p_j}),\ i, j = 1, 2, \ldots, k,\ i < j$. Subsequently, reformulation of these inequalities into the hinge loss form leads to the *relative order* loss function:

$$L_\text{ReO}(\tilde{\boldsymbol{q}}; \mathbb{C}, \mathbf{p}) = \sum_{i=1}^{k} \sum_{j=i}^{k} \big[f(\tilde{\boldsymbol{q}}, \boldsymbol{c}_{p_i}) - f(\tilde{\boldsymbol{q}}, \boldsymbol{c}_{p_j})\big]_+. \tag{1}$$

Given this loss function, the OA can be cast as a constrained optimization problem,

$$\boldsymbol{r}^* = \arg\min_{\boldsymbol{r} \in \Omega_{\boldsymbol{q}}} L_\text{ReO}(\boldsymbol{q} + \boldsymbol{r}; \mathbb{C}, \mathbf{p}), \tag{2}$$

which can be solved with Projected Gradient Descent (PGD) (Madry et al., 2017), an iterative method based on the first-order gradient,

$$\boldsymbol{r}_{t+1} = \text{Clip}_{\Omega_{\boldsymbol{q}}}\big\{\boldsymbol{r}_t - \eta \text{sign}\big[\nabla_{\boldsymbol{r}} L_\text{ReO}(\tilde{\boldsymbol{q}}; \mathbb{C}, \mathbf{p})\big]\big\}, \tag{3}$$

where η is the PGD step size, and $\boldsymbol{r}_0$ is initialized as a zero vector. PGD stops at a predefined maximum iteration T, and the final $\boldsymbol{r}_T$ is the desired adversarial perturbation.

It is worth noting that query image semantics can be drastically changed even with a very slight perturbation (Zhou et al., 2020). As a result, candidates $\mathbb{C}$ may eventually leave the topmost part of the ranking list, which is undesired especially with the "truncated ranking result" constraint as described in Sec.1. To mitigate such side effects, we follow Zhou et al. (2020) and introduce a semantics-preserving term $L_\text{QA+}(\tilde{\boldsymbol{q}}, \mathbb{C})$ to keep $\mathbb{C}$ within the topmost part of the ranking list by raising their *absolute ranks*, *i.e.*, to keep $\boldsymbol{c} \in \mathbb{C}$ ranked ahead of other candidates. Finally, the *relative order* loss term $L_\text{ReO}(\cdot)$ and the *absolute rank* loss term $L_\text{QA+}(\cdot)$ are combined to form the complete OA loss function,

$$L_\text{OA}(\tilde{\boldsymbol{q}}; \mathbb{C}, \mathbf{p}) = L_\text{ReO}(\tilde{\boldsymbol{q}}; \mathbb{C}, \mathbf{p}) + \xi L_\text{QA+}(\tilde{\boldsymbol{q}}, \mathbb{C}), \tag{4}$$

where ξ is a balancing factor between the *relative order* and *absolute rank* goals.

Despite the formulation of Eq. (4), an ideal $\tilde{\boldsymbol{q}}$ that fully satisfy the desired relative order of $\mathbb{C}$ does not necessarily exist. Consider an Euclidean embedding space as an example, where candidates $\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3$ lie consecutively on a straight line. It is impossible to find a query embedding vector that leads to $\boldsymbol{c}_1 \prec \boldsymbol{c}_3 \prec \boldsymbol{c}_2$. That indicates the compatibility between the specified relative order and the factual geometric relations of the candidate embeddings also affects the performance upper-bound of OA. In cases like this, our algorithm can still find an inexact solution that satisfies as many inequalities as possible to approximate the specified *relative order*. In light of this, Kendall's ranking correlation τ between the specified relative order and the real ranking list appears to be a more reasonable performance metric than the success rate for OA. Furthermore, the $\tau_\mathcal{S}$ metric discussed in the following subsection is equivalent to τ in the white-box scenario.

Algorithm 1: SRC: Short-range Ranking Correlation $\tau_{\mathcal{S}}$ for Order Attack.

Input: Selected candidates $\mathbb{C} = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k\}$, permutation vector $\mathbf{p} = [p_1, p_2, \ldots, p_k]$, top-$N$ retrieved candidates $\mathbb{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ *w.r.t* $\tilde{\boldsymbol{q}}$; ($\mathbb{C} \subset \mathbb{D}$, $\mathbb{X} \subset \mathbb{D}$, $N \geqslant k$)
Output: SRC coefficient $\tau_{\mathcal{S}}$
Initialize score matrix $\boldsymbol{S} = \mathbf{0}$ of size $k \times k$; Permuted candidates $\mathbb{C}_{\mathbf{p}} = \{\boldsymbol{c}_{p_1}, \boldsymbol{c}_{p_2}, \ldots, \boldsymbol{c}_{p_k}\}$;
for $i \leftarrow 1, 2, \ldots, k$ **do**
  for $j \leftarrow 1, 2, \ldots, i-1$ **do**
    if $\boldsymbol{c}_i \notin \mathbb{X}$ (i.e. $\nexists m \in \{1, 2, \ldots, N\}$ so that $\boldsymbol{c}_i = \boldsymbol{x}_m$) or $\boldsymbol{c}_j \notin \mathbb{X}$ **then**
      $S_{i,j} = -1$ // out-of-range
    else if $\left[\mathrm{R}_{\mathbb{C}_{\mathbf{p}}}(\boldsymbol{c}_i) > \mathrm{R}_{\mathbb{C}_{\mathbf{p}}}(\boldsymbol{c}_j) \text{ and } \mathrm{R}_{\mathbb{X}}(\boldsymbol{c}_i) > \mathrm{R}_{\mathbb{X}}(\boldsymbol{c}_j)\right]$ or $\left[\mathrm{R}_{\mathbb{C}_{\mathbf{p}}}(\boldsymbol{c}_i) < \mathrm{R}_{\mathbb{C}_{\mathbf{p}}}(\boldsymbol{c}_j) \text{ and } \mathrm{R}_{\mathbb{X}}(\boldsymbol{c}_i) < \mathrm{R}_{\mathbb{X}}(\boldsymbol{c}_j)\right]$ **then**
      $S_{i,j} = +1$ // concordant
    else if $\left[\mathrm{R}_{\mathbb{C}_{\mathbf{p}}}(\boldsymbol{c}_i) > \mathrm{R}_{\mathbb{C}_{\mathbf{p}}}(\boldsymbol{c}_j) \text{ and } \mathrm{R}_{\mathbb{X}}(\boldsymbol{c}_i) < \mathrm{R}_{\mathbb{X}}(\boldsymbol{c}_j)\right]$ or $\left[\mathrm{R}_{\mathbb{C}_{\mathbf{p}}}(\boldsymbol{c}_i) < \mathrm{R}_{\mathbb{C}_{\mathbf{p}}}(\boldsymbol{c}_j) \text{ and } \mathrm{R}_{\mathbb{X}}(\boldsymbol{c}_i) > \mathrm{R}_{\mathbb{X}}(\boldsymbol{c}_j)\right]$ **then**
      $S_{i,j} = -1$ // discordant
return $\tau_{\mathcal{S}} = \sum_{i,j} S_{i,j} / \binom{k}{2} = [\sum_{i,j} S_{i,j}] / [k(k-1)/2]$

2.2 Short-range Ranking Correlation for Black-Box Order Attack

As discussed in Sec.1, black-box OA needs to address multiple challenges including the gradient inaccessibility, query budget limitation, truncated ranking results, and lack of similarity scores. These restrictions collectively invalidate the triplet-style method. To tackle this, we present the "Short-range Ranking Correlation" (SRC, denoted as $\tau_{\mathcal{S}}$) metric, a practical approximation of Kendall's ranking correlation τ (Kendall, 1945) as a surrogate loss function for L_{OA}. It can be optimized with an appropriate black-box optimizer such as NES (Wierstra et al., 2008) and SPSA (Spall et al., 1992), to find an adversarial example as effective as one found by the triplet-style method.

Specifically, to calculate $\tau_{\mathcal{S}}$ given $\mathbb{C}$, $\mathbf{p}$ and the top-N retrieved candidates $\mathbb{X}$ *w.r.t* query $\tilde{\boldsymbol{q}}$, we first initialize a $(k \times k)$-shaped zero matrix $\boldsymbol{S}$, and permute $\mathbb{C}$ into $\mathbb{C}_{\mathbf{p}} = \{\boldsymbol{c}_{p_1}, \boldsymbol{c}_{p_2}, \ldots, \boldsymbol{c}_{p_k}\}$. Assuming $\forall \boldsymbol{c}_i, \boldsymbol{c}_j \in \mathbb{C}_{\mathbf{p}}$ $(i > j, i \neq j)$ exist in $\mathbb{X}$, we define $(\boldsymbol{c}_i, \boldsymbol{c}_j)$ as a *concordant* pair as long as $\mathrm{R}_{\mathbb{C}_{\mathbf{p}}}(\boldsymbol{c}_i)$ and $\mathrm{R}_{\mathbb{X}}(\boldsymbol{c}_i)$ are simultaneously greater or smaller than $\mathrm{R}_{\mathbb{C}_{\mathbf{p}}}(\boldsymbol{c}_j)$ and $\mathrm{R}_{\mathbb{X}}(\boldsymbol{c}_j)$, respectively, where $\mathrm{R}_{\mathbb{X}}(\boldsymbol{c}_i)$ denotes the integer rank value of $\boldsymbol{c}_i$ in $\mathbb{X}$, *i.e.*, $\mathrm{R}_{\mathbb{X}}(\boldsymbol{c}_i) := \arg_m\{\boldsymbol{c}_i = \boldsymbol{x}_m\}$. Otherwise, $(\boldsymbol{c}_i, \boldsymbol{c}_j)$ is defined as a *discordant* pair. A *concordant* pair and a *discordant* pair will be assigned a score of $S_{i,j} = +1$ and $S_{i,j} = -1$, respectively. Apart from that, when $\boldsymbol{c}_i$ or $\boldsymbol{c}_j$ does not exist in $\mathbb{X}$, $S_{i,j}$ will be directly assigned as -1 as a semantics-preserving penalty. In the end, the average score of the lower triangular of $\boldsymbol{S}$ excluding the diagonal is the $\tau_{\mathcal{S}}$, as summarized in Algo. 1. Overall, $\tau_{\mathcal{S}}$ reflects the alignment between the ranking order specified by $\mathbf{p}$ and the order in the real retrieval result, where the semantics-preserving penalty is also incorporated.

The value of $\tau_{\mathcal{S}}$ ranges from -1 to $+1$. When the specified ranking order is fully satisfied, *i.e.*, any pair of $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$ is *concordant*, and none of the elements in $\mathbb{C}$ disappear from $\mathbb{X}$, $\tau_{\mathcal{S}}$ will be 1. In contrast, when every candidate pair is *discordant* or absent from the top-N result $\mathbb{X}$, $\tau_{\mathcal{S}}$ will be -1. Specifically, when $\forall \boldsymbol{c} \in \mathbb{C}$ exists in $\mathbb{X}$, $\tau_{\mathcal{S}}$ degenerates into Kendall's τ between $\mathbf{p}$ and the permutation of $\mathbb{C}$ in $\mathbb{X}$. Namely, $\tau_{\mathcal{S}}$ degenerates gracefully to τ in the white-box scenario.

Although non-differentiable, $\tau_{\mathcal{S}}$ can be optimized by a black-box optimizer to increase the number of concordant pairs, while keeping the candidates within the top-N range as promoted by the semantics-preserving penalty term. Thus, the $\tau_{\mathcal{S}}$ metric can be used as a surrogate objective for black-box OA, *i.e.*, $\boldsymbol{r}^* = \arg\max_{\boldsymbol{r} \in \Omega_{\boldsymbol{q}}} \tau_{\mathcal{S}}(\tilde{\boldsymbol{q}}; \mathbb{C}, \mathbf{p})$, which has a similar effect to the white-box OA.

3 Experiments

To evaluate the white-box and black-box OA, we conduct experiments on the Fashion-MNIST (Xiao et al., 2017) and the Stanford-Online-Products (SOP) (Oh Song et al., 2016) datasets which comprise images of retail commodity. Firstly, we train a CNN with 2-convolution-1-fully-connected network on Fashion-MNIST, and a ResNet-18 (He et al., 2016) without the last fully-connected layer on SOP following Zhou et al. (2020) that focuses on the *absolute rank* attack. Then we perform OA with the corresponding test sets as the candidate database $\mathbb{D}$. Additionally, we also qualitatively evaluate black-box OA on "JD SnapShop" (JingDong, 2020) to further illustrate its efficacy. In our

Table 1: White-box OA on Fashion-MNIST and SOP datasets.

	$k=5$					$k=10$					$k=25$				
ε	0	$\frac{2}{255}$	$\frac{4}{255}$	$\frac{8}{255}$	$\frac{16}{255}$	0	$\frac{2}{255}$	$\frac{4}{255}$	$\frac{8}{255}$	$\frac{16}{255}$	0	$\frac{2}{255}$	$\frac{4}{255}$	$\frac{8}{255}$	$\frac{16}{255}$
Fashion-MNIST $N=\infty$															
$\tau_{\mathcal{S}}$	0.000	0.286	0.412	0.548	0.599	0.000	0.184	0.282	0.362	0.399	0.000	0.063	0.108	0.136	0.149
mR	2.0	4.5	9.1	12.7	13.4	4.5	7.4	10.9	15.2	17.4	12.0	16.1	17.6	18.9	19.4
Stanford Online Products $N=\infty$															
$\tau_{\mathcal{S}}$	0.000	0.396	0.448	0.476	0.481	0.000	0.263	0.348	0.387	0.398	0.000	0.125	0.169	0.193	0.200
mR	2.0	5.6	4.9	4.2	4.1	4.5	12.4	11.2	9.9	9.6	12.0	31.2	28.2	25.5	25.4

Table 2: Searching for balancing factor ξ on both datasets.

ξ	0	10^{-1}	10^{0}	10^{1}	10^{2}	10^{3}	10^{4}	10^{5}	10^{6}	10^{7}
Fashion-MNIST $k=5, N=\infty, \varepsilon=4/255$										
$\tau_{\mathcal{S}}$	0.561	0.467	0.451	*0.412*	0.274	0.052	0.043	0.012	0.007	0.002
mR	27.2	22.7	18.3	*9.1*	4.9	3.2	2.8	2.7	2.7	2.7
Stanford Online Products $k=5, N=\infty, \varepsilon=4/255$										
$\tau_{\mathcal{S}}$	0.932	0.658	0.640	0.634	0.596	*0.448*	0.165	0.092	0.013	0.001
mR	973.9	89.8	48.1	22.4	7.5	*4.9*	2.9	2.8	2.8	2.7

experiments, the value of rank function $\mathrm{R}_{\mathbb{X}}(\cdot)$ starts from 0, *i.e.*, the k-th ranked candidate has the rank value of $k-1$.

Selection of $\mathbb{C}$ and $\mathbf{p}$. As discussed, we assume that the attacker is inclined to select the candidate set $\mathbb{C}$ from the top-N ranked candidates given the visible range limit. For simplicity, we only investigate the (k, N)-OA, *i.e.*, OA with the top-k-within-top-N ($k \leq N$) candidates selected as as $\mathbb{C}$. It is representative because an OA problem with some candidates randomly selected from the top-k results as $\mathbb{C}$ is a sub-problem of (k, N)-OA. Namely, our attack will be effective for any selection of $\mathbb{C}$ as long as the (k, N)-OA is effective. For white-box OA, we conduct experiments with $N=\infty$, and $k \in \{5, 10, 25\}$. For black-box attack, we conduct experiments with $N = \{\infty, 50, k\}$, and $k = \{5, 10, 25\}$. Besides, a random permutation vector $\mathbf{p}$ is used for each individual query.

Evaluation Metric. Since $\tau_{\mathcal{S}}$ is equivalent to τ when $N=\infty$, we use $\tau_{\mathcal{S}}$ as the performance metric for both white-box and black-box OA. Specifically, in each experiment, we conduct $T=10^4$ times of OA attack. In each attack, we randomly draw a sample from $\mathbb{D}$ as the query $\boldsymbol{q}$. In the end, we report the average $\tau_{\mathcal{S}}$ over the T trials. Also, when $N=\infty$, we additionally calculate the mean rank of the candidate set $\mathbb{C}$ (demoted as "mR", which equals $[\sum_i^k \mathrm{R}_{\mathbb{X}}(\boldsymbol{c}_i)]/k$), and report the average mean rank over the T attacks. Larger $\tau_{\mathcal{S}}$ value and smaller mR value are preferable.

Parameter Settings. We set the perturbation budget as $\varepsilon \in \{\frac{2}{255}, \frac{4}{255}, \frac{8}{255}, \frac{16}{255}\}$ for both white-box and black-box attacks. The query budget Q is set to 1.0×10^3. For white-box OA, the PGD step size η is set to $\frac{1}{255}$, the PGD step number to 24. The balancing parameter ξ is set as 10^1 and 10^3 for Fashion-MNIST and SOP respectively. See appendix for the details of the black-box optimizers.

Search Space Dimension Reduction. As a widely adopted trick reported effective in (Dong et al., 2020; Chen et al., 2017; Shukla et al., 2020; Li et al., 2020), we empirically reduce the dimension of adversarial perturbation search space from $(3 \times 224 \times 224)$ to $(3 \times 32 \times 32)$ for black-box OA on Stanford-Online-Products dataset and "JD SnapShop".

3.1 White-Box Order Attack Experiments

The first batch of the experiments is carried out on the Fashion-MNIST dataset, as shown in the upper part of Tab. 1. With the original query image ($\varepsilon = 0$), the expected $\tau_{\mathcal{S}}$ performance of $(5, \infty)$-OA is 0.000, and the $\mathbb{C}$ retains their original ranks as the mR equals 2.0. With a $\varepsilon = 2/255$ adversarial perturbation budget, our OA achieves $\tau_{\mathcal{S}} = 0.286$, which means on average 64.3% [1] of the inequalities reflecting the specified permutations are satisfied by the adversarial example. Meanwhile, the mR changes from 2.0 to 4.5, partly due to adversarial perturbation can move the

[1] Solution of system $(n_{\text{concordant}} - n_{\text{discordant}})/\binom{k}{2} = 0.286$; $(n_{\text{concordant}} + n_{\text{discordant}})/\binom{k}{2} = 1.0$.

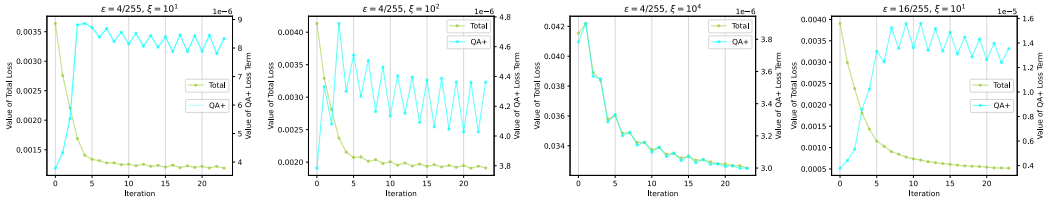

Figure 2: Total loss (left y-axis) and the $L_{\text{QA+}}$ term (right y-axis) during the optimization procedures under different ε and ξ settings on Fashion-MNIST.

query embedding off its original position while seeking for a higher $\tau_{\mathcal{S}}$. Nevertheless, the mR value of 4.5 indicates that the $\mathbb{C}$ are still kept visible in the topmost part of the ranking result by the loss term $L_{\text{QA+}}(\cdot)$. With larger perturbation budget ε, the $\tau_{\mathcal{S}}$ metric increases accordingly, *e.g.*, $\tau_{\mathcal{S}}$ reaches 0.599 when $\varepsilon = 16/255$, which means nearly 80% of the inequalities are satisfied. Likewise, the experimental results on SOP are available in the lower part of Tab. 1, which also demonstrate the effectiveness of our method under different settings.

Besides, we note that different balancing parameter ξ for $L_{\text{QA+}}(\cdot)$ leads to distinct results, as shown in Tab. 2. We conduct $(5, \infty)$-OA with $\varepsilon = 4/255$ with different ξ values ranging from 0 to 10^7 on both datasets. Evidently, a larger ξ leads to a better (smaller) mR value, but meanwhile a worse $\tau_{\mathcal{S}}$ as the weighted $L_{\text{QA+}}(\cdot)$ term dominates the total loss. There is a trade-off between the $\tau_{\mathcal{S}}$ and mR, which is effectively controlled by the tunable constant parameter ξ. Hence, we empirically set ξ as 10^1 and 10^3 for Fashion-MNIST and SOP respectively, in order to keep the mR of most experiments in Tab. 1 below a sensible value, *i.e.*, $50/2$.

Additionally, Tab. 1 reveals that the mR trends w.r.t. ε on the Fashion-MNIST dataset differs from that on the SOP dataset. To investigate this counter-intuitive phenomenon, we plot the loss curves in Fig. 2. In the $\varepsilon = 4/255$, $\xi = 10$ case, the total loss decreases but the $L_{\text{QA+}}$ surges at the beginning and then plateaus. After changing ξ to 100, the $L_{\text{QA+}}$ curve rises more smoothly. The curve eventually decreases at $\xi = 10^4$, along with a small mR and a notable penalty on $\tau_{\mathcal{S}}$ as a result. These figures indicate that the $L_{\text{QA+}}$ term is harder to optimize than the L_{ReO} term on Fashion-MNIST. Besides, the "sawtooth-shaped" $L_{\text{QA+}}$ curves also indicate that the L_{ReO} term is optimized while sacrificing the mR as a side-effect at the even steps, while the optimizer turns to optimize $L_{\text{QA+}}$ at the odd steps due to the semantics-preserving penalty, causing a slight increase in L_{ReO}. This also reveals the optimization difficulty of $L_{\text{QA+}}$. Moreover, comparing the first and the fourth sub-figures, we find a larger perturbation budget ($\varepsilon = 16/255$) not helpful in reducing the optimization difficulty as the $L_{\text{QA+}}$ curve still soars and plateaus. Based on these cues, we speculate that the different curve patterns of mR stem from the optimization difficulty due to *limited search space* (768-dimensional *v.s.* 3072-dimensional), and *different dataset properties*. The intra-class variance of Fashion-MNIST is smaller than that in SOP, which means samples from the same class will be projected into the embedding space close to each other. In this case, it is very difficult to adjust the position of query embedding for a higher $\tau_{\mathcal{S}}$ without sacrificing the mR value. Hence, we conclude that the exact mR value also depends on the dataset property and the relative optimization difficulty of the term L_{ReO} and $L_{\text{QA+}}$ apart from ε. In contrast, the $L_{\text{QA+}}$ term is much easier to optimize in the SOP experiments, possibly due to a larger search space and a larger intra-class variance, hence much more flexibility to simultaneously optimize L_{ReO} and $L_{\text{QA+}}$.

3.2 Black-Box Order Attack Experiments

To simulate the practical black-box attack scenario, we convert the trained ranking models into black-box ones, which only return the top-N candidates without any similarity score. To optimize the surrogate loss $\tau_{\mathcal{S}}$, we adopt and benchmark several black-box optimizers: (1) Random Search (Rand), a baseline method that independently samples every dimension of the perturbation from the uniform distribution $\mathcal{U}(-\varepsilon, +\varepsilon)$; (2) Beta-Attack (Beta), a new black-box method that models the adversarial perturbation with a Beta distribution per dimension, and iteratively adjusts the Beta distribution parameters according to the $\tau_{\mathcal{S}}$ result, which is similar to $\mathcal{N}$-Attack (Li et al., 2019b); (3) Particle Swarm Optimization (PSO) (Shi & Eberhart, 1998), a classic meta-heuristic optimizer; (4) Natural Evolution Strategy (NES) (Ilyas et al., 2018; Wierstra et al., 2008), a method that performs

Table 3: Black-box OA on Fashion-MNIST dataset. In the $N = \infty$ experiments, (τ_S, mR) are reported in each cell, while only τ_S is reported in the cells when N equals 50 or k.

Algorithm	$k = 5$				$k = 10$				$k = 25$			
	$\varepsilon = \frac{2}{255}$	$\varepsilon = \frac{4}{255}$	$\varepsilon = \frac{8}{255}$	$\varepsilon = \frac{16}{255}$	$\varepsilon = \frac{2}{255}$	$\varepsilon = \frac{4}{255}$	$\varepsilon = \frac{8}{255}$	$\varepsilon = \frac{16}{255}$	$\varepsilon = \frac{2}{255}$	$\varepsilon = \frac{4}{255}$	$\varepsilon = \frac{8}{255}$	$\varepsilon = \frac{16}{255}$
None	0.0, 2.0	0.0, 2.0	0.0, 2.0	0.0, 2.0	0.0, 4.5	0.0, 4.5	0.0, 4.5	0.0, 4.5	0.0, 12.0	0.0, 12.0	0.0, 12.0	0.0, 12.0
Fasion-MNIST $N = \infty$												
Rand	0.211, 2.1	0.309, 2.3	0.425, 3.0	0.508, 7.7	0.172, 4.6	0.242, 5.0	0.322, 6.4	0.392, 12.7	0.084, 12.3	0.123, 13.1	0.173, 15.8	0.218, 25.8
Beta	0.241, 2.1	0.360, 2.6	0.478, 4.6	0.580, 19.3	0.210, 4.8	0.323, 5.7	0.430, 9.6	0.510, 30.3	0.102, 12.4	0.163, 13.8	0.237, 19.7	0.291, 42.7
PSO	0.265, 2.1	0.381, 2.3	0.477, 4.4	0.580, 21.1	0.239, 4.8	0.337, 5.7	0.424, 9.7	0.484, 34.0	0.131, 12.7	0.190, 14.6	0.248, 21.7	0.286, 54.2
NES	0.297, 2.3	**0.416, 3.1**	**0.520, 8.7**	**0.630, 46.3**	**0.261, 5.0**	0.377, 6.6	0.473, 14.3	0.518, 55.6	**0.142, 13.0**	0.217, 15.9	0.286, 28.3	0.312, 74.3
SPSA	**0.300, 2.3**	0.407, 3.2	0.465, 7.1	0.492, 16.3	0.249, 5.0	**0.400, 6.6**	**0.507, 12.8**	**0.558, 27.5**	0.135, 12.9	**0.236, 16.3**	**0.319, 27.1**	**0.363, 46.4**
Fashion-MNIST $N = 50$												
Rand	0.207	0.316	0.424	0.501	0.167	0.242	0.321	0.378	0.083	0.123	0.165	0.172
Beta	0.240	0.359	0.470	0.564	0.204	0.323	0.429	0.487	0.103	0.160	0.216	0.211
PSO	0.266	0.377	0.484	0.557	0.239	0.332	0.420	0.458	0.134	0.183	0.220	0.203
NES	**0.297**	**0.426**	**0.515**	**0.584**	**0.262**	0.378	0.463	0.458	**0.141**	0.199	0.223	0.185
SPSA	0.292	0.407	0.468	0.490	0.253	**0.397**	**0.499**	**0.537**	0.131	**0.214**	**0.260**	**0.275**
Fashion-MNIST $N = k$												
Rand	0.204	0.289	0.346	0.302	0.146	0.181	0.186	0.124	0.053	0.062	0.049	0.021
Beta	0.237	0.342	0.372	0.275	0.183	0.236	0.218	0.106	0.072	0.079	0.058	0.020
PSO	0.252	0.342	0.388	0.284	**0.198**	0.240	0.219	0.081	**0.080**	0.082	0.046	0.013
NES	**0.274**	**0.360**	0.381	0.282	**0.198**	0.234	0.213	0.113	0.071	0.076	0.055	0.016
SPSA	**0.274**	**0.360**	**0.412**	**0.427**	0.188	**0.251**	**0.287**	**0.298**	0.067	**0.086**	**0.091**	**0.095**

Projected Gradient Descent using estimated gradient; (5) Simultaneous Perturbation Stochastic Approximation (SPSA) (Uesato et al., 2018; Spall et al., 1992), another iterative method based on estimated gradient, similar to NES. See the appendix for further details about these optimizers.

We first investigate the black-box $(5, \infty)$-OA, as shown in the upper part of Tab. 3 and Tab. 4. In these cases, τ_S does not pose any mR penalty since $N = \infty$. With the Rand optimizer, the τ_S can be optimized to 0.309 in the $\varepsilon = 4/255$ case. As the ε increases, we obtain better τ_S results and larger mR values. Since all the queries of the Rand search are independent, one intuitive way to improve its performance is to leverage the historical query results to adjust the searching pattern.

To this end, we devise the new Beta-Attack that samples perturbations from Beta distributions, and initialize the distribution parameters as 1 where Beta distribution degenerates into Uniform distribution (Rand Search). Thus, Beta-Attack is able to gradually modify its probability density function during the attacking process, so that the next adversarial perturbation drawn from it are possibly more effective. Results in Tab. 3 suggest an evident advantage of Beta compared to Rand, but it turns that such simple parametric distributions are still not enough for modeling the adversarial perturbations. According to the (k, ∞)-OA results, NES and SPSA outperform Rand, Beta and PSO. The τ_S metrics of all algorithms unanimously increase with larger ε, but the side effect of worse (larger) mR is notable, *e.g.*, when $N = 50$ or k, the algorithm may confront with a great penalty due to the absence of the selected candidates from the visible range.

Further results of $(k, 50)$-OA and (k, k)-OA confirms our speculation. When $N = 50$, algorithms that result in a small mR (especially for those with mR $< 50/2$) performs comparably to that in (k, ∞)-OA with the same ε. Algorithms that lead to a large mR with a large ε in (k, ∞)-OA are greatly penalized in $(k, 50)$-OA. This manifests a special characteristic of OA that differs from adversarial attacks in other fields (*e.g.*, classification), τ_S peaks at a small ε, and does not positively correlate with ε like in other attacks. The results of (k, k)-OA conveys similar information, and we note that the black-box $(25, 25)$-OA is extremely difficult.

The optimizers based on estimated gradients perform the best in black-box OA. In difficult cases such as $(25, \infty)$-OA, the black-box optimizer even outperforms the white-box PGD algorithm, possibly due to the built-in randomness of the black-box optimizers enabling better search in the global scope. In contrast, the white-box PGD is more prone to stuck in a local minima.

3.3 Practical Black-Box Order Attack Experiments

The "JD SnapShop" (JingDong, 2020) is an e-commerce platform based on content-based image retrieval (Smeulders et al., 2000). Clients can upload query merchandise images via an `HTTP`-protocol-based API, and the system returns the top-50 similar products. This black-box ranking model exactly matches the setting of $(k, 50)$-OA. Since the system poses an upper limit of 500 queries per day per user, we merely provide a qualitative evaluation of black-box OA.

Table 4: Black-box OA on Stanford Online Product dataset. In the $N = \infty$ experiments, ($\tau_{\mathcal{S}}$, mR) are reported in each cell, while only $\tau_{\mathcal{S}}$ is reported in the cells when N equals 50 or k.

Algorithm	$k=5$				$k=10$				$k=25$			
	$\varepsilon=\frac{2}{255}$	$\varepsilon=\frac{4}{255}$	$\varepsilon=\frac{8}{255}$	$\varepsilon=\frac{16}{255}$	$\varepsilon=\frac{2}{255}$	$\varepsilon=\frac{4}{255}$	$\varepsilon=\frac{8}{255}$	$\varepsilon=\frac{16}{255}$	$\varepsilon=\frac{2}{255}$	$\varepsilon=\frac{4}{255}$	$\varepsilon=\frac{8}{255}$	$\varepsilon=\frac{16}{255}$
None	0.0, 2.0	0.0, 2.0	0.0, 2.0	0.0, 2.0	0.0, 4.5	0.0, 4.5	0.0, 4.5	0.0, 4.5	0.0, 12.0	0.0, 12.0	0.0, 12.0	0.0, 12.0
Stanford Online Product $N=\infty$												
Rand	0.187, 2.6	0.229, 8.5	0.253, 85.8	0.291, 649.7	0.167, 5.6	0.197, 13.2	0.208, 92.6	0.222, 716.4	0.093, 14.1	0.110, 27.6	0.125, 146.7	0.134, 903.7
Beta	0.192, 3.3	0.239, 15.3	0.265, 176.7	0.300, 1257.7	0.158, 6.2	0.186, 19.9	0.207, 139.0	0.219, 992.5	0.099, 15.5	0.119, 37.1	0.119, 206.5	0.132, 1208.5
PSO	0.122, 2.1	0.170, 3.0	0.208, 13.3	0.259, 121.4	0.135, 4.8	0.177, 6.5	0.206, 22.8	0.222, 166.5	0.104, 12.7	0.122, 16.7	0.137, 49.5	0.140, 264.2
NES	**0.254, 3.4**	0.283, 15.6	**0.325, 163.0**	**0.368, 1278.7**	**0.312, 7.2**	**0.351, 26.3**	0.339, 227.1	0.332, 1486.7	**0.242, 18.0**	**0.259, 51.5**	0.250, 324.1	0.225, 1790.8
SPSA	0.237, 3.5	**0.284, 11.9**	0.293, 75.2	0.318, 245.1	0.241, 7.8	0.325, 22.2	**0.362, 112.7**	**0.383, 389.0**	0.155, 18.1	0.229, 41.9	**0.286, 185.6**	**0.306, 557.8**
Stanford Online Product $N=50$												
Rand	0.180	0.216	0.190	0.126	0.163	0.166	0.119	0.055	0.092	0.055	0.016	0.003
Beta	0.181	0.233	0.204	0.119	0.153	0.168	0.116	0.054	0.084	0.057	0.021	0.003
PSO	0.122	0.173	0.183	0.153	0.135	0.164	0.137	0.081	0.093	0.083	0.042	0.011
NES	**0.247**	0.283	0.246	0.152	**0.314**	0.295	0.195	0.077	**0.211**	**0.136**	0.054	0.013
SPSA	0.241	**0.287**	**0.297**	**0.303**	0.233	**0.298**	**0.298**	**0.292**	0.125	0.130	**0.114**	**0.103**
Stanford Online Product $N=k$												
Rand	0.148	0.100	0.087	0.026	0.094	0.044	0.018	0.001	0.023	0.009	0.002	0.001
Beta	0.136	0.106	0.053	0.025	0.076	0.040	0.010	0.004	0.021	0.004	0.001	0.001
PSO	0.102	0.098	0.059	0.031	0.088	0.049	0.022	0.007	0.040	0.015	0.006	0.001
NES	**0.185**	0.139	0.076	0.030	**0.173**	0.097	0.036	0.008	**0.071**	**0.027**	0.007	0.005
SPSA	0.172	**0.154**	**0.141**	**0.144**	0.107	**0.104**	**0.085**	**0.069**	0.026	0.025	**0.017**	**0.016**

As shown in Fig. 1, we select the top-5 candidates as $\mathbb{C}$, and specify $\mathbf{p} = [1, 5, 4, 3, 2]$. By maximizing the $\tau_{\mathcal{S}}$ value with SPSA, we successfully convert the *relative order* to the specified one with $\varepsilon = 1/255$ perturbation. See appendix for more technical details including the product webpages.

4 Related Works

Adversarial Attacks. Szegedy et al. (2013) finds the DNN classifiers susceptible to imperceptible adversarial perturbations. Subsequent works on adversarial attack (Dong et al., 2020) can be categorized into several groups: **(1)** white-box attack, which assumes the model details including the gradient are accessible (Goodfellow et al., 2014; Kurakin et al., 2016; Madry et al., 2017; Moosavi-Dezfooli et al., 2016; Carlini & Wagner, 2017; Athalye et al., 2018; 2017; Croce & Hein, 2020); **(2)** transfer-based attack, which are based on the transferability of adversarial examples (Dong et al., 2018; Xie et al., 2019; Dong et al., 2019a); **(3)** score-based attack, which only depends on the soft classification labels, *i.e.*, the logit values (Ilyas et al., 2018; Uesato et al., 2018; Li et al., 2019b; Chen et al., 2017; Andriushchenko et al., 2019). Ilyas et al. (2018) propose a black-box threat model for classification that is similar to black-box ranking threat model; and **(4)** decision-based attack, a type of attack that requires the least amount of information from the model, *i.e.*, the hard label (one-hot classification result) (Brendel et al., 2018; Chen & Jordan, 2019; Dong et al., 2019b; Shukla et al., 2020; Li et al., 2020). These extensive adversarial attack studies unanimously focus on classification, which means they are not directly suitable for ranking.

Adversarial Ranking. The existence of the vulnerability in DNN classifiers inspired attacks against deep ranking models, but this topic has not yet been sufficiently explored. Judging on the purpose of manipulating the ranking list, we propose a new taxonomy for attacks against deep ranking – *absolute rank* attacks and *relative order* attacks. Most *absolute rank* attacks attempt to induce random "misranking" (Tolias et al., 2019; Li et al., 2019a; Liu et al., 2019; Yang et al., 2018; Zhao et al., 2019; Wang et al., 2020; Zheng et al., 2018; Bouniot et al., 2020; Feng et al., 2020). There are also works that aim to incur purposeful changes in *absolute rank*, *i.e.*, to raise or lower the ranks of specific candidates (Bai et al., 2019; Zhou et al., 2020). For example, the $L_{\text{QA+}}$ (Zhou et al., 2020) adopted in this paper is a typical *absolute rank* attack. On the contrary, the *relative order* attacks remains under-explored. This is the first work that tampers the relative order in deep ranking.

5 Conclusion

Deep ranking systems have inherited the adversarial vulnerabilities of deep neural networks. In this paper, we propose a new adversarial attack named *Order Attack*, which manipulates the *relative order* among selected candidates. Extensive experimental evaluations of the white-box and black-box *Order Attack* illustrate their effectiveness, as well as the deep ranking systems' vulnerability to the *Order Attack*. In future works, we plan to explore more efficient *Order Attack* algorithms with better gradient estimation, and more robust ranking models resistant to *Order Attack*.

REFERENCES

Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. *arXiv preprint arXiv:1912.00049*, 2019.

Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*, 2017.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

Song Bai, Yingwei Li, Yuyin Zhou, Qizhu Li, and Philip HS Torr. Metric attack and defense for person re-identification. *arXiv preprint arXiv:1901.10650*, 2019.

Quentin Bouniot, Romaric Audigier, and Angelique Loesch. Vulnerability of person re-identification models to metric adversarial attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *ArXiv*, abs/1712.04248, 2018.

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

Jianbo Chen and Michael I Jordan. Boundary attack++: Query-efficient decision-based adversarial attack. *arXiv preprint arXiv:1904.02144*, 2019.

Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 15–26, 2017.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *arXiv preprint arXiv:2003.01690*, 2020.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *CVPR*, June 2018.

Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *CVPR*, pp. 4312–4321, 2019a.

Yinpeng Dong, Hang Su, Baoyuan Wu, Zhifeng Li, Wei Liu, Tong Zhang, and Jun Zhu. Efficient decision-based black-box adversarial attacks on face recognition. *CVPR*, pp. 7706–7714, 2019b.

Yinpeng Dong, Qi-An Fu, Xiao Yang, Tianyu Pang, Hang Su, Zihao Xiao, and Jun Zhu. Benchmarking adversarial robustness on image classification. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Yan Feng, Bin Chen, Tao Dai, and Shutao Xia. Adversarial attack on deep product quantization network for image retrieval. *arXiv preprint arXiv:2002.11374*, 2020.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, June 2016.

Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. *arXiv preprint arXiv:1804.08598*, 2018.

JingDong. *SnapShop API*, 2020. URL `https://neuhub.jd.com/ai/api/image/snapshop`.

Maurice G Kendall. The treatment of ties in ranking problems. *Biometrika*, 33(3):239–251, 1945.

James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pp. 1942–1948. IEEE, 1995.

Sungyeon Kim, Minkyo Seo, Ivan Laptev, Minsu Cho, and Suha Kwak. Deep metric learning beyond binary supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2288–2297, 2019.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, pp. 1097–1105, 2012.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

Huichen Li, Xiaojun Xu, Xiaolu Zhang, Shuang Yang, and Bo Li. Qeba: Query-efficient boundary-based blackbox attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1221–1230, 2020.

Jie Li, Rongrong Ji, Hong Liu, Xiaopeng Hong, Yue Gao, and Qi Tian. Universal perturbation attack against image retrieval. In *ICCV*, pp. 4899–4908, 2019a.

Yandong Li, Lijun Li, Liqiang Wang, Tong Zhang, and Boqing Gong. Nattack: Learning the distributions of adversarial examples for an improved black-box attack on deep neural networks. *arXiv preprint arXiv:1905.00441*, 2019b.

Zhuoran Liu, Zhengyu Zhao, and Martha Larson. Who's afraid of adversarial queries?: The impact of image modifications on content-based image retrieval. In *ICMR*, pp. 306–314. ACM, 2019.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *CVPR*, pp. 2574–2582, 2016.

Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *CVPR*, pp. 4004–4012, 2016.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pp. 815–823, 2015.

Yuhui Shi and Russell Eberhart. A modified particle swarm optimizer. In *1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)*, pp. 69–73. IEEE, 1998.

Satya Narayan Shukla, Anit Kumar Sahu, Devin Willmott, and J Zico Kolter. Hard label black-box adversarial attacks in low query budget regimes. *arXiv preprint arXiv:2007.07210*, 2020.

Arnold WM Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on pattern analysis and machine intelligence*, 22(12):1349–1380, 2000.

James C Spall et al. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE transactions on automatic control*, 37(3):332–341, 1992.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Giorgos Tolias, Filip Radenovic, and Ondrej Chum. Targeted mismatch adversarial attack: Query with a flower to retrieve the tower. In *ICCV*, pp. 5037–5046, 2019.

Jonathan Uesato, Brendan O'Donoghue, Aaron van den Oord, and Pushmeet Kohli. Adversarial risk and the dangers of evaluating against weak attacks. *arXiv preprint arXiv:1802.05666*, 2018.

Hongjun Wang, Guangrun Wang, Ya Li, Dongyu Zhang, and Liang Lin. Transferable, controllable, and inconspicuous adversarial attacks on person re-identification with deep mis-ranking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning fine-grained image similarity with deep ranking. In *CVPR*, pp. 1386–1393, 2014.

Daan Wierstra, Tom Schaul, Jan Peters, and Juergen Schmidhuber. Natural evolution strategies. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 3381–3387. IEEE, 2008.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L Yuille. Improving transferability of adversarial examples with input diversity. In *CVPR*, pp. 2730–2739, 2019.

Erkun Yang, Tongliang Liu, Cheng Deng, and Dacheng Tao. Adversarial examples for hamming space search. *IEEE transactions on cybernetics*, 2018.

Guoping Zhao, Mingyu Zhang, Jiajun Liu, and Ji-Rong Wen. Unsupervised adversarial attacks on deep feature-based retrieval with gan. *arXiv preprint arXiv:1907.05793*, 2019.

Zhedong Zheng, Liang Zheng, Zhilan Hu, and Yi Yang. Open set adversarial examples. *arXiv preprint arXiv:1809.02681*, 2018.

Mo Zhou, Zhenxing Niu, Le Wang, Zhanning Gao, Qilin Zhang, and Gang Hua. Ladder loss for coherent visual-semantic embedding. *arXiv preprint arXiv:1911.07528*, 2019.

Mo Zhou, Zhenxing Niu, Le Wang, Qilin Zhang, and Gang Hua. Adversarial ranking attack and defense. *arXiv preprint arXiv:2002.11293*, 2020.

A VISUALIZATION OF BLACK-BOX ORDER ATTACK

A.1 MORE TECHNICAL DETAILS OF OA AGAINST "JD SNAPSHOP"

In this subsection we present some additional technical details about the e-commerce platform (API) "JD SnapShop" (JingDong, 2020) and the Fig. 1:

1. In Fig. 1 the "Han Chinese Clothing" query image is of the standard $(3 \times 224 \times 224)$ size.
2. Since the perturbation tensor contains negative values, we normalize it before displaying:
$$\text{Normalize}(\boldsymbol{r}) = 0.5 + 0.5 * \boldsymbol{r} / \max\big(\text{abs}(\boldsymbol{r})\big). \tag{5}$$
3. The perturbation budget in the case of Fig. 1 is $\varepsilon = 1/255$, but after saving the adversarial image into the JPEG/PNG file, the infinite norm of the absolute difference between $\tilde{\boldsymbol{q}}$ and $\boldsymbol{q}$ may slightly exceed the ε budget due to the lossy compression algorithm of JPEG/PNG. See the next subsection for more details on the ε selection.
4. The perturbation has a much lower resolution than the query because we have reduced the search space dimension from $(3 \times 224 \times 224)$ to $(3 \times 32 \times 32)$, which is a trick widely used in the literature. See Sec. C.2 for the ablation study.
5. The selected candidates C are [A, B, C, D, E]. Since the permutation vector is $\mathbf{p} = [1, 5, 4, 3, 2]$, the expected relative order among $\mathbb{C}$ is $C_{p_1} \prec C_{p_2} \prec C_{p_3} \prec C_{p_4} \prec C_{p_5}$, *i.e.*A $\prec$ E $\prec$ D $\prec$ C $\prec$ B.
6. Each product corresponds to multiple images. The displayed candidate images are their default product image.
7. The API in fact provides the similarity scores for every candidate. Indeed these discriminative information can be leveraged for, *e.g.*better gradient estimation by using scores other than $\{+1, -1\}$ in $\boldsymbol{S}$ when calculating $\tau_{\boldsymbol{S}}$ (See Algo. 1), but the other practical ranking applications may not necessarily provide them. Hence, we simply ignore the similarity information during the attacking process, deliberately increasing the difficulty.
8. In Fig. 1, the original similarity scores of candidates from A to E are [0.7132, 0.6336, 0.6079, 0.5726, 0.5700]. With our adversarial query, the scores of A to E are changed into [0.6960, 0.5724, 0.5763, 0.5827, 0.5898].
9. The API supports a "`topK`" argument, which enables us to change the visible range N. We leave it as the recommended default value 50.
10. From Fig. 1, we notice some visually duplicated images among the candidates. For instance, there are many other candidates similar to candidate E, due to reasons such as different sellers reusing the same image. These images are not adjacent to each other in the ranking list, since the platform assigns them with different similarity scores. For instance, the 5-th and 8-th candidates in the first row of Fig. 1 are assigned with similarity scores $[0.5700, 0.5521]$, while the 2-nd, 7-th, and 10-th items in the second row with similarity scores $[0.5898, 0.5728, 0.5536]$. Whether the calculation of similarity scores involves multiple cues (*e.g.*by aggregating the similarity scores of multiple product images) or even engineering tricks are unknown and beyond the scope of our discussion.
11. Users (without any business contract) are only allowed to perform 500 times of queries per day.
12. The API documentation can be found at
`https://aidoc.jd.com/image/snapshop.html`.
13. The SKU ID atop of every candidate image can be used to browse the real product webpage on the "JingDong" shopping platform. The URL format is
`https://item.jd.com/<SKU-ID>.html`
For example, the webpage for the product with SKU ID `72210617020` is located at
`https://item.jd.com/72210617020.html`
Note, due to irresistible reasons such as some sellers withdrawing their product and the service provider updating their ranking algorithm and database, some of the links may become invalid during the review process, and the ranking result for the same query may change. Source code of our implementation is provided as supplementary material. The source code is intended for ACADEMIC PURPOSE ONLY [2].

[2]The anonymous authors are not responsible for any consequence of abusing the provided code.

A.1.1 EMPIRICAL & QUALITATIVE PARAMETER SEARCH FOR ε

Since the perturbation budget ε affects the performance of OA and the adversarial perturbation imperceptibility, we also search for a proper ε for the OA against "JD SnapShop".

Due to the limitation that only 500 times of queries per day are allowed for each user, we merely present an empirical and qualitative parameter search for ε. As shown in Tab. 5, we test the "JD SnapShop" model with adversarial query images modified by the Rand algorithm using different ε values, and conduct 50 times of attack per value. Our qualitative observation has been summarized in the table.

Table 5: Empirical & Qualitative Parameter Search for ε on "JD SnapShop".

ε	Observation
$16/255$	Almost any $\forall \boldsymbol{c} \in \mathbb{C}$ disappear from the top-N result.
$8/255$	In most cases only $0 \sim 1$ selected candidate remains within the top-N.
$4/255$	In most cases only $1 \sim 3$ selected candidates remain within the top-N.
$2/255$	Nearly all $\boldsymbol{c} \in \mathbb{C}$ remain in top-N with significant order change. (suitable for $\mathbf{p}$ with $p_1 \neq 1$)
$1/255$	Rank of top-1 candidate seldom changes. The rest part of the list has been slightly changed. (suitable for $\mathbf{p}$ with $p_1 = 1$)

From the table, we find that $\varepsilon = 1/255$ and $\varepsilon = 2/255$ are the most suitable choices for the $(5, 50)$-OA against "JD SnapShop". This is meanwhile very preferable since such slight perturbations are imperceptible to human. As shown in Fig. 3, the $\varepsilon = 1/255, 2/255, 4/255$ adversarial perturbation can hardly be perceived.

$\varepsilon = 0$ $\varepsilon = 1/255$ $\varepsilon = 2/255$ $\varepsilon = 4/255$ $\varepsilon = 8/255$ $\varepsilon = 16/255$

Figure 3: Imperceptibility: Images perturbed under different perturbation budgets.

A.1.2 MORE SHOWCASES ON "JD SNAPSHOP"

- In showcase #2 (Fig.4), the original similarity scores of the top-5 candidates are [0.7551, 0.6586, 0.6586, 0.6533, 6507]. They are changed into [0.6748, 0.6723, 0.6723, 0.6921, 0.6609] with the perturbation.
- In showcase #3 (Fig.5), the original top-5 candidate similarity scores [0.8522, 0.8333, 0.8341, 0.7659, 0.7159] have been changed into [0.8792, 0.7928, 0.8470, 0.7958, 0.7130] with the perturbation.
- Fig. 6 shows two examples where one selected candidate disappear from the top-N results with the adversarial query. In the "white shoes" case, the top-5 candidate similarity scores have been changed from [0.8659, 0.8653, 0.8648, 0.8619, 0.8603] to [N/A, 0.8689, 0.8603, 0.8640, 0.8641]. In the "vase" case, the top-5 candidate similarity scores have been changed from [0.9416, 0.9370, 0.9360, 0.9350, 0.9349] to [N/A, 0.9338, 0.9427, 0.9392, 0.9361].
- Fig. 7 shows some long-tail queries on which our OA will not take effect, because a large portion of the top-ranked candidates have completely the same similarity score. In the first row, *i.e.*results for a "Machine Learning" textbook query, the similarity score of the 5-th to 7-th candidates are 0.9242. The score of the 9-th to 11-th candidates are 0.9240. That of the 22-th to 50-th are the same 0.9202525. In the "Deep Learning" textbook case (the second row), the similarity score of the 27-th to 50-th candidates are 0.9288423. In the "RTX 3090 GPU" case (the third row), the similarity score of the 7-th to 9-th candidate are unexceptionally 0.5581. Our OA cannot change the *relative order* among those candidates with completely the same similarity score.

Figure 4: Showcase #2: “Red wind coat” query image with $\varepsilon = 2/255$ perturbation.

Figure 5: Showcase #3: “iPhone” query image with $\varepsilon = 1/255$ perturbation.

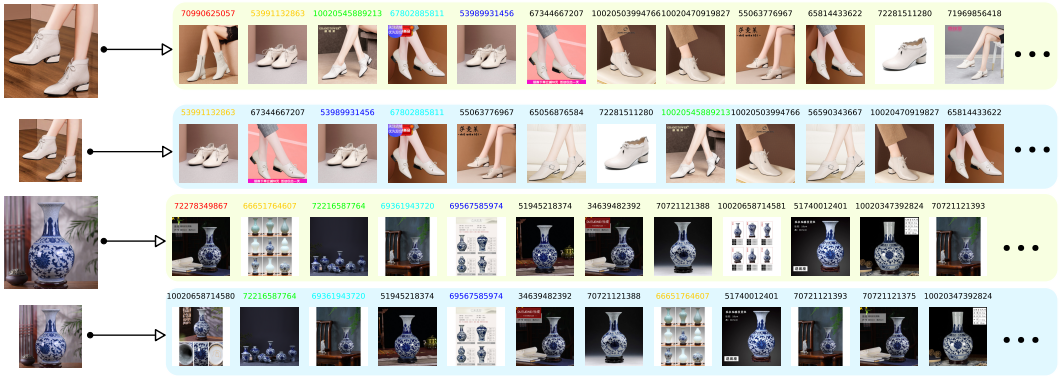

Figure 6: Two cases where the top-1 ranked candidate disappeared from the top part.

Figure 7: Three long-tail cases where OA will not be effective.

A.2 ON FASHION-MNIST & STANFORD-ONLINE-PRODUCTS

In this subsection we present some visualizations of the black-box OA on the Fashion-MNIST dataset and the SOP dataset, as shown in (Fig. 8, Fig. 9, Fig. 10) and (Fig. 11, Fig. 12, Fig. 13), respectively. The adversarial perturbations in all these figures are found under the $\varepsilon = 4/255$ budget. The N is 50 for these results.

All these figures are picture matrices of size $(4, 2 + 2k)$. Picture $(1, 3)$, $(1, 5)$ and $(1, 7)$ are the original query, the perturbation and the perturbed query image, respectively. The second row in each figure is the original query and the corresponding ranking list. The third row in each figure is the permuted top-k candidates. The forth row in each figure is the adversarial query and the corresponding ranking list. Every picture has been annotated with its ID in the dataset and the label for classification.

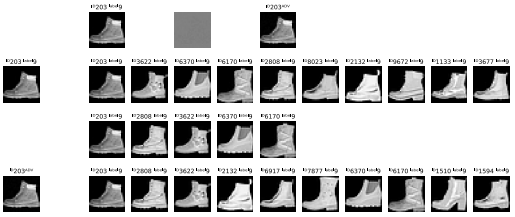

Figure 8: Fashion #1. $k = 5$, $\tau_S = 1.0$

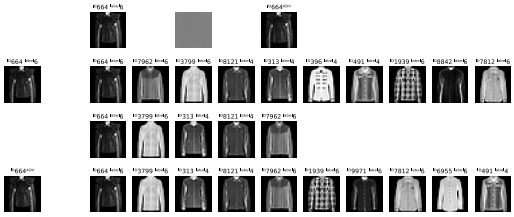

Figure 9: Fashion #2. $k = 5$, $\tau_S = 1.0$

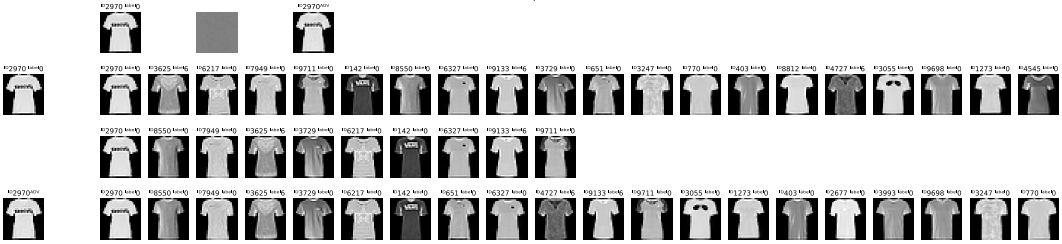

Figure 10: Fashion #3. $k = 10$, $\tau_S = 1.0$

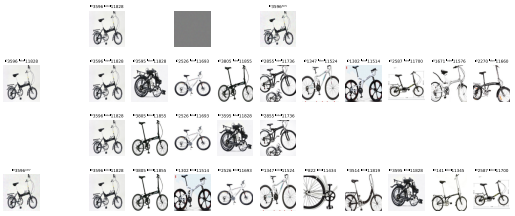

Figure 11: SOP #1. $k = 5$, $\tau_S = 1.0$

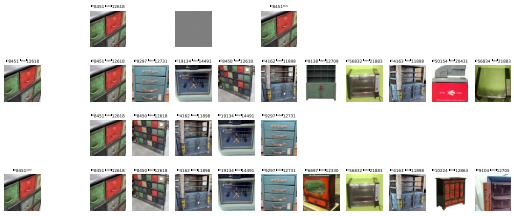

Figure 12: SOP #2. $k = 5$, $\tau_S = 1.0$

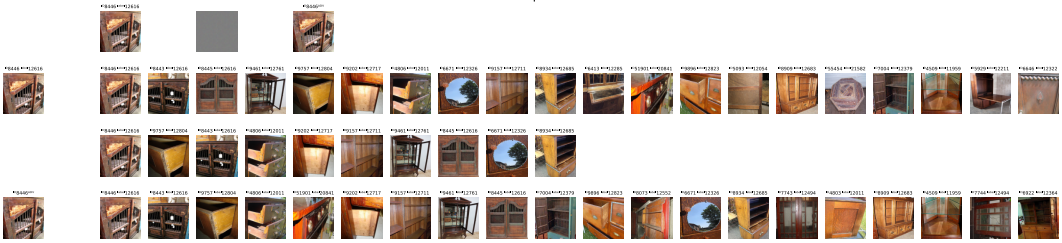

Figure 13: SOP #3. $k = 10$, $\tau_S = 0.96$

B ADDITIONAL EXPERIMENTS & DISCUSSIONS ON WHITE-BOX OA

B.1 ABLATION STUDY: SEMANTICS-PRESERVING TERM $L_{\text{QA+}}$

Table 6: Ablation Study: White-Box OA with $\xi = 0$ (*i.e.*without the $L_{\text{QA+}}$ term).

	$k=5$					$k=10$					$k=25$				
ε	0	$\frac{2}{255}$	$\frac{4}{255}$	$\frac{8}{255}$	$\frac{16}{255}$	0	$\frac{2}{255}$	$\frac{4}{255}$	$\frac{8}{255}$	$\frac{16}{255}$	0	$\frac{2}{255}$	$\frac{4}{255}$	$\frac{8}{255}$	$\frac{16}{255}$
Fashion-MNIST $\xi = 0$															
$\tau_{\mathcal{S}}$	0.000	0.336	0.561	0.777	0.892	0.000	0.203	0.325	0.438	0.507	0.000	0.077	0.131	0.170	0.189
mR	2.0	5.5	27.2	52.7	75.6	4.5	8.0	17.3	40.4	63.4	12.0	16.4	19.2	22.8	25.3
Stanford Online Products $\xi = 0$															
$\tau_{\mathcal{S}}$	0.000	0.932	0.970	0.975	0.975	0.000	0.632	0.760	0.823	0.832	0.000	0.455	0.581	0.646	0.659
mR	2.0	973.9	1780.1	2325.5	2421.9	4.5	1222.7	3510.2	5518.6	6021.4	12.0	960.4	2199.2	3321.4	3446.3

As shown in Tab. 6, after removing the $L_{\text{QA+}}$ term from the loss function (*i.e.*setting $\xi = 0$), the white-box OA can achieve a better $\tau_{\mathcal{S}}$, meanwhile a worse mR. When comparing it with Tab. 1, we find that (1) the semantics-preserving term $L_{\text{QA+}}$ is effective for keeping the selected C within top-N results; (2) $L_{\text{QA+}}$ will increase the optimization difficulty, so there will be a trade off between L_{ReO} and $L_{\text{QA+}}$. These results support our discussion in Sec. 3.1.

B.2 LOSS CURVES ON THE SOP DATASET

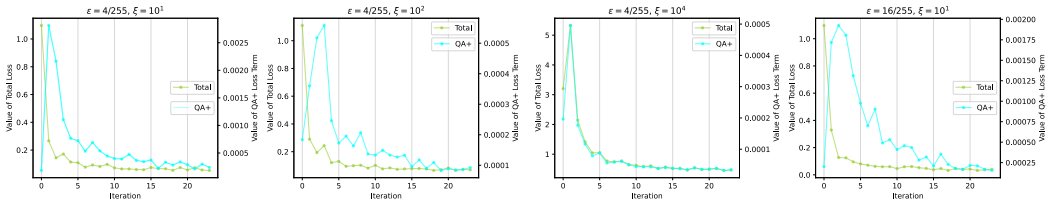

Figure 14: Curves of the total loss and the $L_{\text{QA+}}$ term during the optimization process under different ε and ξ settings on SOP. Left y-axis denotes the value of total loss, while the right y-axis denotes the value of $L_{\text{QA+}}$ term.

To further support our discussion in Sec. 3.2, we also plot the loss curves on the SOP dataset like those on the Fashion-MNIST dataset (see Fig. 2). As shown in Fig. 14, the $L_{\text{QA+}}$ is relatively easy to optimize on the SOP dataset even if the ξ is not set to a very large value (*i.e.*$\xi = 10$). Comparing the first and the last plot, we find that with a larger perturbation budget $\varepsilon = 16/255$, the value of $L_{\text{QA+}}$ becomes smaller. These cues can help interpret the difference in the growth pattern of mR in Tab. 1.

B.3 DISCUSSION: "RANDOM START" TRICK FOR PGD

It is noted that Madry et al. (2017) used a "random start" trick, which means to initialize the adversarial perturbation $\boldsymbol{r}$ as a random vector within $\Omega_{\mathbf{p}}$ instead of a zero vector before the Projected Gradient Descent (PGD). This trick is harmful for Order Attack.

A random start may drastically change the query semantics and push the query embedding off its original position (Zhou et al., 2020). In this case the expectation of the mR value is much larger (worse) than that with a zero start. As a result, the optimizer would waste more iterations to minimize the unnecessary mR penalty, which may even fail eventually.

The "random start" trick can be even more harmful in the black-box scenario, as NES (Ilyas et al., 2018) and SPSA (Uesato et al., 2018) are two methods that performs Projected Gradient Descent with estimated gradients. With a random perturbation as the start, all the selected candidates may disappear from the top-N result before the first iteration, resulting in $\tau_{\mathcal{S}} = -1$. Once all samples drew from the neighbor area of the random start leads to $\tau_{\mathcal{S}} = -1$, the optimization will fail due to the estimated gradient being completely invalid (zero gradient).

C ADDITIONAL EXPERIMENTS & DISCUSSIONS ON BLACK-BOX OA

C.1 DISCUSSION: QUERY BUDGET Q AND τ_S

Table 7: τ_S with different query budget Q.

Algorithm	Fashion-MNIST $N = \infty, k = 5, \varepsilon = \frac{4}{255}$				
	$Q = 10^2$	5×10^2	$*10^3$	5×10^3	10^4
Rand	0.233, 2.2	0.291, 2.2	0.309, 2.3	0.318, 2.2	0.320, 2.2
Beta	0.249, 2.2	0.313, 2.4	0.360, 2.6	0.368, 2.6	0.382, 2.4
PSO	0.280, 2.6	0.341, 2.4	0.381, 2.3	0.382, 2.4	0.385, 2.4
NES	0.309, 2.6	0.380, 2.9	0.416, 3.1	0.431, 2.9	0.438, 2.9
SPSA	0.292, 2.6	0.365, 2.8	0.407, 3.2	0.421, 2.9	0.433, 2.8

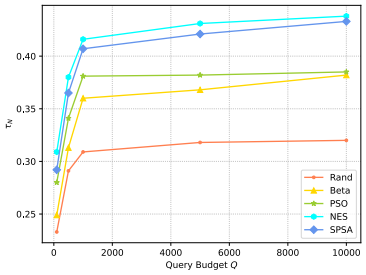

Figure 15: τ_S with different Q.

In this subsection, we study the influence of the query budget Q on the τ_S performance. As shown in Tab. 7 and Fig. 15, the performance curves of all black-box optimizers are positively correlated with the query budget Q, but will eventually plateau. Note, in our context the Q is not a tunable hyper-parameter of the optimizers, but a constant for simulating the black-box scenario.

C.2 ABLATION STUDY: SEARCH SPACE DIMENSION REDUCTION TRICK

Table 8: Ablation study of the Dimension Reduction (DR) trick for black-box OA with SOP dataset.

Algorithm	$N = \infty\ k = 5$				$N = 50\ k = 5$				$N = 5\ k = 5$			
	$\varepsilon = \frac{2}{255}$	$\frac{4}{255}$	$\frac{8}{255}$	$\frac{16}{255}$	$\frac{2}{255}$	$\frac{4}{255}$	$\frac{8}{255}$	$\frac{16}{255}$	$\frac{2}{255}$	$\frac{4}{255}$	$\frac{8}{255}$	$\frac{16}{255}$
None	0.0, 2.0	0.0, 2.0	0.0, 2.0	0.0, 2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Rand (w/o DR)	0.106, 2.1	0.151, 3.1	0.190, 13.1	0.224, 117.5	0.101	0.139	0.167	0.148	0.076	0.086	0.058	0.026
Rand (w/ DR)	0.187, 2.6	0.229, 8.5	0.253, 85.8	0.291, 649.7	0.180	0.216	0.190	0.126	0.148	0.100	0.087	0.026
Beta (w/o DR)	0.120, 2.2	0.158, 3.8	0.199, 21.8	0.231, 205.7	0.115	0.164	0.173	0.141	0.097	0.096	0.060	0.035
Beta (w/ DR)	0.192, 3.3	0.239, 15.3	0.265, 176.7	0.300, 1257.7	0.181	0.233	0.204	0.119	0.136	0.106	0.053	0.025
PSO (w/o DR)	0.128, 2.1	0.174, 3.1	0.219, 13.0	0.259, 122.0	0.133	0.175	0.199	0.155	0.097	0.095	0.060	0.036
PSO (w/ DR)	0.122, 2.1	0.170, 3.0	0.208, 13.3	0.259, 121.4	0.122	0.173	0.183	0.153	0.102	0.098	0.059	0.031
NES (w/o DR)	0.139, 2.3	0.192, 4.8	0.244, 31.9	0.266, 300.0	0.128	0.192	0.208	0.166	0.108	0.102	0.079	0.039
NES (w/ DR)	0.254, 3.4	0.283, 15.6	0.325, 163.0	0.368, 1278.7	0.247	0.283	0.246	0.152	0.185	0.139	0.076	0.030
SPSA (w/o DR)	0.135, 2.4	0.171, 3.9	0.209, 15.9	0.226, 45.2	0.140	0.176	0.205	0.223	0.108	0.110	0.146	0.143
SPSA (w/ DR)	0.237, 3.5	0.284, 11.9	0.293, 75.2	0.318, 245.1	0.241	0.287	0.297	0.303	0.172	0.154	0.141	0.144

In this subsection, we study the effectiveness of the dimension reduction trick, which has been widely adopted in the literature (Dong et al., 2020; Chen et al., 2017; Shukla et al., 2020; Li et al., 2020). As shown in Tab. 8, all black-box optimizers benefit from this trick except for PSO, as illustrated by the performance gains. We leave the analysis on the special characteristics of PSO for future work.

D DETAILS OF BLACK-BOX OPTIMIZATION ALGORITHMS

In this section we present the algorithm details for (1) Random Search (Rand); (2) Beta-Attack (Beta); (3) Particle Swarm Optimization (PSO) Shi & Eberhart (1998); (4) Natural Evolution Strategy (NES) Ilyas et al. (2018); Wierstra et al. (2008); and (5) Simultaneous Perturbation Stochastic Approximation (SPSA) Uesato et al. (2018); Spall et al. (1992).

D.1 RANDOM SEARCH (RAND)

As a baseline algorithm for black-box optimization, Random Search assumes each element in the adversarial perturbation to be *i.i.d*, and samples each element from the uniform distribution within $\Omega_{\boldsymbol{p}}$, namely $\boldsymbol{r} = [r_1, r_2, \ldots, r_D]$ where $r_i \sim \mathcal{U}(-\varepsilon, +\varepsilon)$ $(i = 1, 2, \ldots, D)$ in each iteration. The best historical result is the output of the algorithm, as summarized in Algo.2. This algorithm is free of hyper-parameters.

This algorithm never stuck at the local-maxima, which means it has a great ability to search for solutions from the global scope. But its drawback is also clear, as each trial of this algorithm is independent to each other. In our implementation, we conduct OA on a batch of random perturbations to accelerate the experiments, with the batch size set as $H = 50$.

Algorithm 2: RandSearch: The naïve random search algorithm.

Input: Query Image $\boldsymbol{q}$, Query Budget Q, Selected Candidates $\mathbb{C}$, Permutation vector $\mathbf{p}$
Output: Adversarial Query $\tilde{\boldsymbol{q}}$
Initialize $\tilde{\boldsymbol{q}} \leftarrow \boldsymbol{q}$, and the best score $s^* \leftarrow \tau_{\mathcal{S}}(\tilde{\boldsymbol{q}})$;
for $i \leftarrow 1, 2, \ldots, Q$ **do**
  Sample $\boldsymbol{r} \sim \mathcal{U}^D(-\varepsilon, +\varepsilon)$;
  if $\tau_{\mathcal{S}}\Big(\text{Clip}_{\Omega_{\boldsymbol{p}}}(\boldsymbol{p} + \boldsymbol{r})\Big) > s^*$ **then**
    $\tilde{\boldsymbol{q}} \leftarrow \text{Clip}_{\Omega_{\boldsymbol{p}}}(\boldsymbol{p} + \boldsymbol{r})$ `// update the best q`
    $s^* \leftarrow \tau_{\mathcal{S}}\Big(\text{Clip}_{\Omega_{\boldsymbol{p}}}(\boldsymbol{p} + \boldsymbol{r})\Big)$ `// update the best s`
return $\tilde{\boldsymbol{q}}$

D.2 BETA-ATTACK (BETA)

Beta-Attack is similar to $\mathcal{N}$-Attack (Li et al., 2019b), but a notable difference is that the Gaussian distributions in $\mathcal{N}$-Attack are replaced with Beta distributions. We choose Beta distribution because the shape of its probability density function is much more flexible than that of the Gaussian distribution, which may be beneficial for modeling the adversarial perturbations. Besides, according to our observation, $\mathcal{N}$-Attack is too prone to stuck at local maxima for OA, leading to a low $\tau_{\mathcal{S}}$ performance.

The key idea of Beta-Attack is to find the parameters for a parametric distribution $\pi(\boldsymbol{z}|\theta)$ (θ denotes the parameters) from which the adversarial perturbations drawn from it is likely effective. And $\pi(\boldsymbol{z}|\theta)$ is a combination of D independent Beta distributions[3], with parameter $\theta = [\boldsymbol{a}; \boldsymbol{b}]$ where $\boldsymbol{a} = [a_1, a_2, \ldots, a_D]$, $\boldsymbol{b} = [b_1, b_2, \ldots, b_D]$, and $z_i \sim \text{Beta}(a_i, b_i) \in [0, 1]$, $(i = 1, 2, \ldots, D)$. Let $\mathcal{T}(\boldsymbol{z}) = \tau_{\mathcal{S}}\big(\text{Clip}_{\Omega_{\boldsymbol{q}}}(\boldsymbol{q} + \varepsilon(2\boldsymbol{z} - 1))\big)$, where $\varepsilon(2\boldsymbol{z} - 1) = \boldsymbol{r}$ is the adversarial perturbation, we hope to maximize the mathematical expectation of $\mathcal{T}(\boldsymbol{z})$ over distribution $\pi(\boldsymbol{z}|\theta)$:

$$\max_{\theta} \mathbb{E}_{\pi(\boldsymbol{z}|\theta)}\Big[\mathcal{T}(\boldsymbol{z})\Big] := \int \mathcal{T}(\boldsymbol{z}) \cdot \pi(\boldsymbol{z}|\theta) d\boldsymbol{z} \tag{6}$$

[3]The multivariate Beta distribution, *a.k.a* Dirichlet distribution $\boldsymbol{z} \sim \text{Dir}(\boldsymbol{a})$ is not used here because its $\sum z_i = 1$ restriction further shrinks the search space hence may lower the upper-bound of $\tau_{\mathcal{S}}$.

The gradient of the expectation with respect to θ is

$$\nabla_\theta \mathbb{E}_{\pi(\boldsymbol{z}|\theta)}\Big[\mathcal{T}(\boldsymbol{z})\Big] = \nabla_\theta \int \mathcal{T}(\boldsymbol{z}) \cdot \pi(\boldsymbol{z}|\theta) d\boldsymbol{z} \tag{7}$$

$$= \int \mathcal{T}(\boldsymbol{z}) \cdot \frac{\pi(\boldsymbol{z}|\theta)}{\pi(\boldsymbol{z}|\theta)} \nabla_\theta \pi(\boldsymbol{z}|\theta) d\boldsymbol{z} \tag{8}$$

$$= \int \pi(\boldsymbol{z}|\theta) \cdot \mathcal{T}(\boldsymbol{z}) \cdot \nabla_\theta \log\big[\pi(\boldsymbol{z}|\theta)\big] d\boldsymbol{z} \tag{9}$$

$$= \mathbb{E}_{\pi(\boldsymbol{z}|\theta)}\Big[\mathcal{T}(\boldsymbol{z}) \cdot \nabla_\theta \log\big[\pi(\boldsymbol{z}|\theta)\big]\Big] \tag{10}$$

where

$$\nabla_\theta \log\big[\pi(\boldsymbol{z}|\theta)\big] = \Big[\nabla_{\boldsymbol{a}} \log\big[\pi(\boldsymbol{z}|\theta)\big];\ \nabla_{\boldsymbol{b}} \log\big[\pi(\boldsymbol{z}|\theta)\big]\Big], \tag{11}$$

$$\nabla_{\boldsymbol{a}} \log\big[\pi(\boldsymbol{z}|\theta)\big] = \psi^{(0)}(\boldsymbol{a}+\boldsymbol{b}) - \psi^{(0)}(\boldsymbol{a}) + \log(\boldsymbol{z}), \tag{12}$$

$$\nabla_{\boldsymbol{b}} \log\big[\pi(\boldsymbol{z}|\theta)\big] = \psi^{(0)}(\boldsymbol{a}+\boldsymbol{b}) - \psi^{(0)}(\boldsymbol{b}) + \log(1-\boldsymbol{z}), \tag{13}$$

and $\psi^{(n)}(z)$ is the n-th derivative of the digamma function. The Eq. 7 to Eq. 13 means that the gradient of the expectation of $\mathcal{T}(\boldsymbol{z})$ with respect to θ can be estimated by approximating the expectation in Eq. 10 with its mean value using a batch of random vectors, *i.e.*,

$$\mathbb{E}_{\pi(\boldsymbol{z}|\theta)}\Big[\mathcal{T}(\boldsymbol{z}) \cdot \nabla_\theta \log\big[\pi(\boldsymbol{z}|\theta)\big]\Big] \approx \frac{1}{H}\sum_{i=1}^{H}\Big[\mathcal{T}(\boldsymbol{z}_i) \cdot \nabla_\theta \log\big[\pi(\boldsymbol{z}_i|\theta)\big]\Big] \tag{14}$$

where H denotes the batchsize, and $\boldsymbol{z}_i$ is drawn from $\pi(\boldsymbol{z}|\theta)$. Thus, the parameters θ of the Beta distributions can be updated with Stochastic Gradient Ascent, *i.e.*

$$\theta_{t+1} \leftarrow \theta_t + \eta \nabla_\theta \mathbb{E}_{\pi(\boldsymbol{z}|\theta)}\Big[\mathcal{T}(\boldsymbol{z})\Big], \tag{15}$$

where η is a constant learning rate for the parameters θ. With a set of trained parameters θ, we expect a higher $\tau_{\mathcal{S}}$ performance from a random perturbation $\boldsymbol{z}$ drawn from $\pi(\boldsymbol{z}|\theta)$.

In our experiments, we initialize $\boldsymbol{a} = 1$, and $\boldsymbol{b} = 1$. This is due to a very important property of Beta distribution that it degenerates into Uniform distribution when $a = 1$ and $b = 1$. Namely, our Beta-Attack is initialized as the "Rand Search", but it is able to update its parameters according to the historical $\tau_{\mathcal{S}}$ performance, changing the shape of its probability density function in order to improve the expectation of the $\tau_{\mathcal{S}}$ of the next adversarial perturbation drawn from it. Besides, the batch size H is set to 50 for all the experiments.

As discussed in Sec. 3.2, and shown in Tab. 3 and Tab. 4, the Beta-Attack obviously outperforms the "Rand Search", and is comparable to PSO, but is still surpassed by the NES and SPSA methods. We speculate that such simple parametric distributions are not enough for modeling the adversarial perturbations for the challenging OA problem. Due to its performance not outperforming PSO, NES and SPSA, the new Beta-Attack is not regarded as a major contribution of our paper, but its comparison with the other methods are very instructive.

D.2.1 Parameter Search for Beta-Attack

Table 9: Parameter search of learning rate η. Fashion-MNIST, $N = \infty$, $k = 5$, $\varepsilon = \frac{4}{255}$

Learning Rate η	0.0	1.5	*3.0	4.5	6.0
SRC $\tau_{\mathcal{S}}$	0.290, 2.2	0.332, 2.4	0.360, 2.6	0.341, 2.5	0.330, 2.5

We conduct parameter search of η on the Fashion-MNIST, as shown in Tab. 9. From the table, we find that the $\tau_{\mathcal{S}}$ performance peaks at $\eta = 3.0$, so we set this value as the default learning rate for the experiments on Fashion-MNIST. Apart from that, we empirically set $\eta = 0.5$ for the experiments on SOP following a similar parameter search.

D.3 PARTICLE SWARM OPTIMIZATION (PSO)

Particle Swarm Optimization (Kennedy & Eberhart, 1995; Shi & Eberhart, 1998) is a classical meta-heuristic optimization method. Let constant H denote the population (swarm) size, we randomly initialize the H particles (vectors) as $\mathbb{Y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_H\}$. The positions of these particles are iteratively updated according to the following velocity formula:

$$\boldsymbol{v}_i \leftarrow \omega \boldsymbol{v}_i + \phi_p \cdot \text{rand}() \cdot (\boldsymbol{p}_i - \boldsymbol{y}_i) + \phi_g \cdot \text{rand}() \cdot (\boldsymbol{g} - \boldsymbol{y}_i) \tag{16}$$

$$\boldsymbol{y}_i \leftarrow \boldsymbol{y}_i + \boldsymbol{v}_i \tag{17}$$

where $\boldsymbol{y}_i \in \mathbb{Y}$, rand() generates a random number within the interval $[0, 1]$, ω denotes the inertia (momentum), $\boldsymbol{p}_i$ is the historical best position of particle i, $\boldsymbol{g}$ is the global historical best position among all particles, ϕ_p and ϕ_g are two constant parameters. As a meta-heuristic method, PSO does not guarantee that a satisfactory solution will eventually be discovered.

In the implementation, we directly represent the adversarial query $\tilde{\boldsymbol{q}}$ withe the particles. And we also additionally clip the particles as

$$\boldsymbol{y}_i \leftarrow \min\left\{\boldsymbol{q} + \varepsilon, \max\left\{\boldsymbol{q} - \varepsilon, \boldsymbol{y}_i\right\}\right\} \tag{18}$$

at the end of each PSO iteration, which is the only difference of our implementation compared to the standard PSO (Shi & Eberhart, 1998). In all the experiments, the swarm size is set as $H = 40$.

D.3.1 PARAMETER SEARCH FOR PSO

Table 10: Parameter search for PSO. Fashion-MNIST, $N = \infty$, $k = 5$, $\varepsilon = \frac{4}{255}$

Inertia ω	0.8	1.0	*1.1	1.2	1.4
SRC $\tau_{\mathcal{S}}$	0.321, 2.3	0.363, 2.3	0.381, 2.3	0.369, 2.4	0.349, 2.4
ϕ_p	0.37	0.47	*0.57	0.67	0.77
SRC $\tau_{\mathcal{S}}$	0.353, 2.3	0.375, 2.3	0.381, 2.3	0.376, 2.3	0.364, 2.3
ϕ_g	0.24	0.34	*0.44	0.54	0.64
SRC $\tau_{\mathcal{S}}$	0.353, 2.3	0.372, 2.3	0.381, 2.3	0.380, 2.3	0.359, 2.3

We conduct parameter search of ω, ϕ_p and ϕ_g for PSO on the Fashion-MNIST dataset, as shown in Tab. 10.

- Inertia ω: This parameter affects the convergence of the algorithm. A large ω endows PSO with better global searching ability, while a small ω allows PSO to search better in local areas. From the table, we find the PSO performance peaks at $\omega = 1.1$, which means the global searching ability is relatively important for solving the black-box Order Attack problem.
- Constants ϕ_p and ϕ_g: These parameter control the weights of a particle's historical best position ("the particle's own knowledge") and the swarm's historical global best position ("the knowledge shared among the swarm") in the velocity formula. With a relatively small ϕ_p, and a relatively large ϕ_g, the swarm will converge faster towards the global best position, taking a higher risk of being stuck at a local maxima. From the table, we note that both constants should be kept relatively small, which means it is not preferred to converge too fast towards either the particles' individual best positions or the global best position, for sake of better global searching ability of the algorithm.

We conclude from the parameter search that the global searching ability is important for solving the black-box OA, as reflected by the parameters. Hence, we empirically set $\omega = 1.1$, $\phi_p = 0.57$, and $\phi_g = 0.44$ for PSO in all experiments.

D.4 NATURAL EVOLUTIONARY STRATEGY (NES)

NES (Ilyas et al., 2018) is based on Wierstra et al. (2008), which aims to find the adversarial perturbation through projected gradient method (Madry et al., 2017) with estimated gradients. Specifically, let the adversarial query $\tilde{\boldsymbol{q}}$ be the mean of a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{z}|\tilde{\boldsymbol{q}}, \Sigma)$

where the covariance matrix Σ is a hyper-parameter matrix, and $\mathcal{T}(\boldsymbol{z}) = \tau_{\mathcal{S}}\big(\text{Clip}_{\Omega_{\boldsymbol{q}}}(\tilde{\boldsymbol{q}} + \boldsymbol{z})\big)$, NES first estimates the gradient of the expectation:

$$\nabla_{\tilde{\boldsymbol{q}}}\mathbb{E}_{\mathcal{N}(\boldsymbol{z}|\tilde{\boldsymbol{q}},\Sigma)}[\mathcal{T}(\boldsymbol{z})] = \mathbb{E}_{\mathcal{N}(\boldsymbol{z}|\tilde{\boldsymbol{q}},\Sigma)}\Big\{\mathcal{T}(\boldsymbol{z})\nabla_{\tilde{\boldsymbol{q}}}[\log\mathcal{N}(\boldsymbol{z}|\tilde{\boldsymbol{q}},\Sigma)]\Big\} \tag{19}$$

$$\approx \frac{1}{H}\sum_{i=1}^{H}\Big\{\mathcal{T}(\boldsymbol{z}_i)\nabla_{\tilde{\boldsymbol{q}}}[\log\mathcal{N}(\boldsymbol{z}_i|\tilde{\boldsymbol{q}},\Sigma)]\Big\}. \tag{20}$$

$$\tag{21}$$

The batch of $\boldsymbol{z}_i$ are sampled from $\mathcal{N}(\tilde{\boldsymbol{q}},\Sigma)$. Then NES updates the adversarial example $\tilde{\boldsymbol{q}}$ (*i.e.*the mean of the multivariate Gaussian) with projected gradient ascent (Madry et al., 2017):

$$\tilde{\boldsymbol{q}}_{t+1} \leftarrow \text{Clip}_{\Omega_{\boldsymbol{q}}}\Big\{\tilde{\boldsymbol{q}}_t + \eta\cdot\text{sign}\big(\nabla_{\tilde{\boldsymbol{q}}}\mathbb{E}_{\mathcal{N}(\boldsymbol{z}|\tilde{\boldsymbol{q}},\Sigma)}[\mathcal{T}(\boldsymbol{z})]\big)\Big\}, \tag{22}$$

where η is a constant learning rate. Following Ilyas et al. (2018), we set the covariance matrix as a scaled identity matrix, *i.e.*$\Sigma = \sigma I$. The batch size is set to $H = 50$ for all experiments.

D.4.1 Parameter Search for NES

Table 11: Parameter Search for NES. Fashion-MNIST, $N = \infty$, $k = 5$, $\varepsilon = \frac{4}{255}$

Learning Rate η	$\frac{1}{255}$	$*\frac{2}{255}$	$\frac{3}{255}$	$\frac{4}{255}$	$\frac{5}{255}$
SRC $\tau_{\mathcal{S}}$	0.404, 3.0	0.416, 3.1	0.394, 2.8	0.398, 2.8	0.387, 2.8
σ	ε/0.125	ε/0.25	$*\varepsilon$/0.5	ε/1.0	ε/2.0
SRC $\tau_{\mathcal{S}}$	0.403, 2.9	0.407, 2.9	0.416, 3.1	0.400, 2.9	0.382, 2.8

As shown in Tab. 11, the $\tau_{\mathcal{S}}$ performance of NES peaks at $\eta = 2/255$ and $\sigma = \varepsilon/0.5$. So we use this setting for all the rest experiments with NES.

D.5 Simultaneous Perturbation Stochastic Approximation (SPSA)

SPSA (Uesato et al., 2018) is based on Spall et al. (1992). In particular, the implementation of (Uesato et al., 2018) is very similar to the NES implementation discussed above. The only difference is that, in (Uesato et al., 2018), the random vectors used for estimating the gradients are sampled from Rademacher distributions (*i.e.*Bernoulli ± 1) instead of the Gaussian distributions: $\boldsymbol{z} = \delta\boldsymbol{u} = \delta[u_1, u_2, \ldots, u_D]$, and $\forall i \in 1, \ldots, D,\ u_i \sim \text{Rademacher}()$, where δ is a tunable parameter. Apart from that, we also set the batch size as $H = 50$ for all SPSA experiments.

Compared to the NES algorithm in the experiments, we speculate that such random vector sampling strategy provides SPSA better ability to jump out from local maxima due to a larger norm of the random perturbations. As a result, SPSA performs better than NES in some cases.

D.5.1 Parameter Search for SPSA

Table 12: Parameter Search for SPSA. Fashion-MNIST, $N = \infty$, $k = 5$, $\varepsilon = \frac{4}{255}$

Learning Rate η	$\frac{1}{255}$	$*\frac{2}{255}$	$\frac{3}{255}$	$\frac{4}{255}$	$\frac{5}{255}$
SRC $\tau_{\mathcal{S}}$	0.383, 3.0	0.407, 3.2	0.374, 2.8	0.360, 2.8	0.365, 2.8
Perturbation size δ	$\frac{1}{255}$	$*\frac{2}{255}$	$\frac{3}{255}$	$\frac{4}{255}$	$\frac{5}{255}$
SRC $\tau_{\mathcal{S}}$	0.322, 2.7	0.407, 3.2	0.401, 2.9	0.400, 2.9	0.397, 2.8

According to the parameter search in Tab. 12, we set $\eta = 2/255$ and $\delta = 2/255$ as the default parameter in all other experiments with SPSA.