Machine Learning (2024) 113:1705–1723
https://doi.org/10.1007/s10994-022-06252-2

DIMBA: discretely masked black-box attack in single object tracking

Xiangyu Yin[1] · **Wenjie Ruan**[1] · **Jonathan Fieldsend**[1]

Received: 31 May 2022 / Revised: 9 August 2022 / Accepted: 19 September 2022 /
Published online: 31 October 2022

Abstract
The adversarial attack can force a CNN-based model to produce an incorrect output by craftily manipulating human-imperceptible input. Exploring such perturbations can help us gain a deeper understanding of the vulnerability of neural networks, and provide robustness to deep learning against miscellaneous adversaries. Despite extensive studies focusing on the robustness of image, audio, and NLP, works on adversarial examples of visual object tracking—especially in a black-box manner—are quite lacking. In this paper, we propose a novel adversarial attack method to generate noises for single object tracking under black-box settings, where perturbations are merely added on initialized frames of tracking sequences, which is difficult to be noticed from the perspective of a whole video clip. Specifically, we divide our algorithm into three components and exploit reinforcement learning for localizing important frame patches precisely while reducing unnecessary computational queries overhead. Compared to existing techniques, our method requires less time to perturb videos, but to manipulate competitive or even better adversarial performance. We test our algorithm in both long-term and short-term datasets, including OTB100, VOT2018, UAV123, and LaSOT. Extensive experiments demonstrate the effectiveness of our method on three mainstream types of trackers: discrimination, Siamese-based, and reinforcement learning-based trackers. We release our attack tool, DIMBA, via GitHub https://github.com/TrustAI/DIMBA for use by the community.

Keywords Visual object tracking · Adversarial example · Black-box attack

Editors: Yu-Feng Li, Prateek Jain.

✉ Wenjie Ruan
w.ruan@exeter.ac.uk

Xiangyu Yin
xy329@exeter.ac.uk

Jonathan Fieldsend
J.E.Fieldsend@exeter.ac.uk

[1] College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter EX4 4QF, UK

1 Introduction

While deep learning has achieved a breakthrough in solving the problems that have been experienced by the artificial intelligence and machine learning community over the past decade, several studies have revealed that Deep Neural Networks (DNNs) are vulnerable to adversarial perturbations (Goodfellow et al., 2015) on image processing tasks (Moosavi-Dezfooli et al., 2016; Szegedy et al., 2014; Xie et al., 2017). For images, such perturbations are often too small to be perceptible, yet they can completely fool a DNN classifier, detector, or segmentation analyzer, causing them to predict incorrect categories or contours. This leads to great concerns under the circumstances where deep learning models are deployed rapidly in safety and security-critical applications in particular, e.g., self-driving cars, surveillance, drones, and robotics (Mnih et al., 2015). Besides the computer vision applications, recent works also investigate adversarial attacks on other tasks, e.g. natural language processing (Zhang et al., 2019a), audio recognition (Yakura & Sakuma, 2019), and malware detection (Grosse et al., 2017).

Single object tracking(SOT), as one of the fundamental problems in computer vision, has recently experienced tremendous improvement through DNNs and plays a significant role in practical security applications such as self-driving systems, robotics, etc., Mnih et al. (2015). In terms of the tracking procedure, it can be mainly divided into three categories, Siamese-based trackers (Bertinetto et al., 2016; Li et al., 2018; Zhang et al., 2019b; Zhu et al., 2018), discrimination trackers (Danelljan et al., 2019, 2020), and reinforcement learning-based trackers (Yun et al., 2017). Siamese-based trackers define the tracking problem as a one-stage detection problem and locate the object on subsequent frames that have the most similar feature representations with the initial template, their reliance on initialized frames especially targeted regions is fully exploited in our proposed algorithm. In contrast, discrimination trackers predict object locations based on two sub-modules, which are target classification and target estimation. The third category, reinforcement learning-based trackers, formulate the whole tracking procedure as a Markov Decision Process and select different actions according to the agent state at the current step. In recent years, after the concept of adversarial attack was proposed by Szegedy et al. (2014), intensive follow-up methods were inspired to demonstrate various adversaries to deceive deep learning models (Goodfellow et al., 2015; Kurakin et al., 2017; Madry et al., 2019), adversarial attacks concerning visual object tracking have also been explored by plenty of works. For example, Yan et al. (2020a) has proposed a Cooling-Shrinking Loss to train the perturbation generator to achieve an effective and efficient adversarial attacking algorithm. Moreover, spatial-temporal sparse noise was applied in Guo et al. (2020) along targeted or untargeted trajectories. By categorizing the tracking problem into classification and regression branches, Chen et al. (2020) focused on free-model object tracking with dual attention.

Whereas, current attack algorithms applied on SOT exhibit several limitations that may severely restrict their generality in practice. Specifically, we highlight the following disadvantages: (1) *Most tracking adversaries cannot be extended to black-box SOT applications*. Given comprehensive knowledge of model architecture and parameters, miscellaneous approaches are capable of generating effective perturbations over the whole video clip based on the computation of network gradient. However, the target network is often inaccessible within safety-critical scenarios where we can only obtain hard-label predictions during the whole tracking procedure. Therefore, practical black-box attack algorithms are worthy of exploration. (2) *Current methods compose perturbations often on multiple frames.* As illustrated above, existing white-box attacks can realize powerful overall results, but most of them are derived from

noises attached to a large portion of frames. Although the initial frame of a video plays a vital role in SOT, few works pay attention to this, either in white-box or black-box scenarios. For instance, the Hijacking algorithm (Yan et al., 2020b) generates an adversary on a special clip of the video, and the IoU attack (Jia et al., 2021) proposes a continuous black-box attack framework imposed from the 2_{nd} frame to N_{th} frame. (3) *Recent query-based black-box attacks applied on SOT do not consider computational efficiency*. As far as we know, none of the existing query-based black-box attacks on SOT considers query efficiency. Jia et al. (2021) focuses on temporal correlations between adjacent frames. The gradually increasing perturbation magnitude can surely influence the tracking performance, but its effectiveness heavily relies on query times for each frame and the randomness of the Gaussian distribution.

Overall, different from black-box attacks on image classification or segmentation where perturbations are merely added to a single picture, the tracking performance in SOT is determined by the whole video clip, and as the number of perturbed frames increases, adversaries will be detected more easily. Meanwhile, the gradient information is completely lacking within black-box scenarios. It seems that a sacrifice of query times is unavoidable to improve adversarial results in a query-based black-box attack. Therefore, we propose a question:

Can we combine efficiency and effectiveness in black-box attack on SOT?

Or in other words, can we select the most fragile part of a video, and realize heavily shifted tracking results more quickly? In this paper, we propose the *Di*screte *M*asked *B*lack-Box *a*ttack (DIMBA) algorithm, which is mainly inspired by mechanisms of SiamRPN-based trackers that achieve the balance between speed and performance based on initialized frames and generalized to other types of trackers. In contrast to previous works, we firstly introduce the decision-based attack strategy by crafting heavy perturbations on significant regions in the initial frame, then remove unnecessary noises and decrease the adversarial magnitude using a zeroth-order optimization algorithm. In summary, the key contributions of our paper are as follows:

(1) We formulate the query-based black-box attack problem on SOT in a query-efficient manner. Compared to recursively generated perturbations in each frame, we only focus on significant regions in the initial frame, and firstly introduce a decision-based attack strategy in adversarial SOT problems.
(2) To reduce unnecessary patch-based heavy perturbations on specific areas in initialized frames, and increase the probability of generating perturbations causing similar attack performance within a smaller perturbing radius, we introduce a novel grid searching strategy.
(3) The comprehensively devised experiments over OTB100, UAV123, LaSOT, and VOT2018 datasets show that DIMBA attack can generate perturbations more efficiently, and achieve competitive or even better performance compared to SOTA black-box attacks on SOT.

2 Related works

2.1 Adversarial attacks on visual object tracking

Wide applications of visual object tracking have led to numerous specialized real-world techniques, which have also resulted in well-crafted attacks from the adversarial

perspective. Taking the realm of physical world attacks into account, Eykholt et al. (2018) analyzed adversarial stickers on stop signs in the context of autonomous driving to fool YOLO (Redmon et al., 2016). Jia et al. (2019) proposed a 'tracking hijacking' technique to fool multiple object trackers with imperceptible perturbations computed for object detectors in the perceptual pipeline of autonomous driving. Meanwhile, Yan et al. (2020a) developed an attacking technique to deceive single object trackers based on SiamRPN++ (Li et al., 2018). Their method trains a generator model to construct adversarial frames under a 'cooling-shrinking' loss, which is manipulated to cool down the hot target regions and force the bounding boxes to shrink during online tracking. Huang et al. (2020) delved into physical attacks on object detectors in the wild by developing a universal camouflage for object categories. One-shot attack (Chen et al., 2020) demonstrated the possibility to craft adversaries in the first frame of a video clip, forcing trackers, especially SiamRPN-based ones to lose the target in subsequent frames. A spatial-aware attack (SPARK) is proposed in Guo et al. (2020) to fool online trackers. This approach imposes an L_p constraint over perturbations while computing them incrementally based on previous frames. Extensive experiments show that their adversaries are capable of fooling multiple state-of-the-art trackers.

Different from the above methods proposed in white-box settings, Jia et al. (2021) explores the black-box attack by utilizing temporal correspondence between adjacent frames and incrementally adding noises from the second frame to subsequent frames. From the perspective of attack strategies, however, it focuses on locally anchored noises between adjacent templates and relies excessively on the successful randomness of perturbations in earlier frames due to the temporal momentum, which in essence sacrifices the efficiency for generality. Therefore in this paper, we make full use of the prior knowledge of search regions in the initial frame, especially existing in SiamRPN-based trackers, to improve the query efficiency: Formulating tracking as a one-shot detection problem, SiamRPN-based trackers aim at locating objects that have similar appearance with the initial template on the search region in each frame. Though search regions are not considered in reinforcement learning-based trackers, the initial frame plays an important role as the starting point of iterative actions in RNN-based frameworks. Equivalently in discriminative tracking processes, the target classification and location regression module can be impacted by attached perturbations surrounding the object on the first frame.

In Table 1, we compare our proposed method with previous attack algorithms from different perspectives, including the knowledge of perturbed models, number of frames under adversarial attacks, transferability of adversaries between different trackers, and whether or not the proposed algorithm is a decision-based one.

2.2 Deep reinforcement learning

Due to its ability to scale to previously intractable decision-making problems, Deep Reinforcement Learning (DRL) has been a growing area recently. Kickstarting this revolution (Mnih et al., 2015), for example, firstly learns to play a range of Atari 2600 video games at a superhuman level directly from pixel-level knowledge, whilst demonstrating that RL agents could be trained on raw, high-dimensional observations based on reward signals. As another standout success, AlphaGo (Silver et al., 2016) parallelled the historic achievement of IBM's Deep Blue and defeated a human world champion in Go.

Over time, several types of RL algorithms have been introduced and they can be divided into three groups: Actor-Only, Critic-Only, and Actor-Critic methods. Policy gradient

Table 1 A high-level comparison with previous attack methods on visual object tracking

Methods	Black-box	Single-frame	Transferability	Decision-based
Hijacking (Jia et al., 2019)	✗	✓	✗	✗
UP (Ding et al., 2021)	✗	✗	✗	✗
One-Shot (Chen et al., 2020)	✗	✓	✓	✗
Spark (Guo et al., 2020)	✗	✗	✓	✗
IoU (Jia et al., 2021)	✓	✗	✗	✗
Proposed attack	✓	✓	✗	✓

methods such as REINFORCE algorithms (Williams, 1992) are chiefly Actor-Only and optimized over a large set of parameterized policies. In contrast, Critic-Only methods including Q-learning (Watkins & Dayan, 1992) and SARSA (Sutton & Barto, 2018) approximate solutions to the Bellman equation and learn the optimal value functions. To combine the advantages of Actor-Only and Critic-Only methods, Actor-Critic methods generate continuous actions step by step, while the large variance in the policy gradients of an Actor is reduced by a Critic.

3 Methodology

In this section, we first introduce the preliminaries of our proposed attack method. As shown in Fig. 2, The general pipeline of our algorithm consists of three parts. Firstly, We introduce a momentum-based as well as a patch-based perturbation generation process to accumulate heavily perturbed frames as candidate examples. Then a key-patch selection module divides the object-surrounding noise into different regions and computes the importance for each of them so that we can remove less important patches step by step and remain the approximately same attack results within a bounded range. At last, an iterative boundary-walking strategy is utilized to compress perturbation magnitude while maintaining attack results within a specific region. Perturbed by our method, Fig. 1 quantitatively illustrates IoU scores with the increase of frame indexes For simplicity, only One Pass Evaluation (OPE) is considered in the following sections.

3.1 Preliminaries

We denote a video sample by $v \in \mathcal{V} \subset \mathbb{R}^{N\times H\times W\times C}$ with N, H, W, C referring to the number of frames, height, width, and the number of channels respectively. A specific frame can be denoted as $v_i (i \in 1, \dots N)$, where N is the length of video v. Generally, SOT learns a tracking model $\mathcal{T}(v;\theta) : \mathcal{V} \rightarrow \Leftarrow\mathcal{B}\Leftrightarrow\mathcal{S}\Rightarrow$ by minimizing regression loss between ground truth and predicted bounding boxes in each frame and maximizing similarity of predicted bounding boxes between adjacent frames. $\mathcal{B} \in \mathcal{R}^{(N-1)\times 4}$ indicates localizing matrix, where each row $[x_i, y_i, w_i, h_i]$ denotes the x-axis and y-axis coordinates, width, and height of the predicted bounding box for v_i(The initialized frame and its ground truth bounding box are prior knowledge). Meanwhile, $\mathcal{S}$ collects the highest confidence scores for each frame. According to the evaluation method, SOT can be summarized into two categories. The first

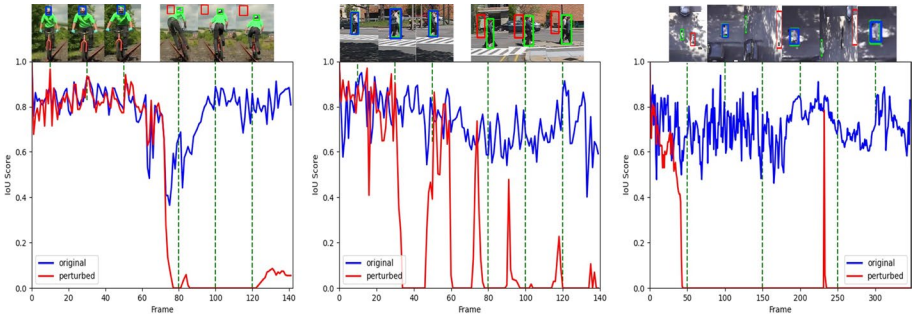

Fig. 1 Visualization of tracking results generated by trackers from three different tracking categories under DIMBA Attack, including SiamRPN++ (Li et al., 2019) (left), ADNet (Yun et al., 2017) (middle), and PrDiMP50 (Danelljan et al., 2020) (right). Clipped frames above the chart qualitatively demonstrate the behaviors of trackers with or without attack. Green bounding boxes refer to ground truths, blue ones measure original tracking results, and red ones illustrate failed tracking performance. The charts below indicate IoU scores between predicted bounding boxes and ground truths, and the tracking performance with or without attack is separately represented in red and blue lines (Color figure online)

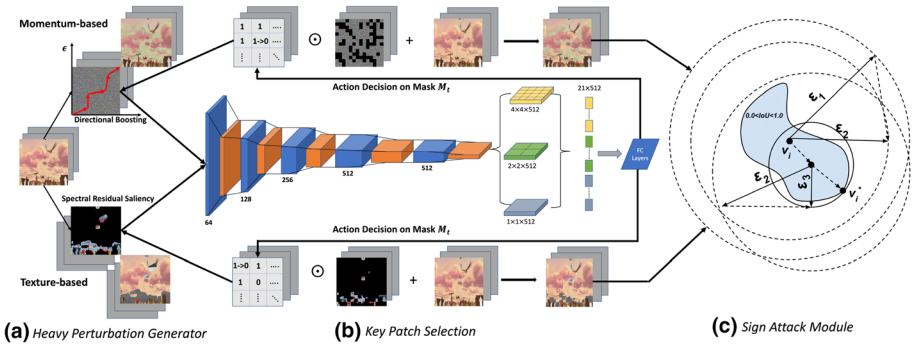

Fig. 2 Overview of DIMBA framework, which contains heavy perturbation generator, key patch selection, and sign attack module, **a** Heavy Perturbation Generator initially constructs candidate adversarial videos, originating from either momentum-based approach or patch-based approach. **b** Key Patch Selection assigns the mask value of heavily perturbed patches to be 0 based on an Actor-Critic network, of which structure is proposed above. **c** Sign Attack Module estimates gradients around designated directions calculated from previous steps and compresses adversarial magnitude while maintaining attack results within a specific region

one initializes only once in a single video, which is also called One Pass Evaluation (OPE). In contrast, the second approach can restart the tracker several frames after the failed one, such as testing trackers on Visual Object Tracking Challenge 2018 (Kristan, 2018). The goal of an adversarial attack in SOT is to find an adversarial example v^* that can fool the network to make a shifted or even target-lost bounding box in the sequence, while keeping v^* within the ϵ-ball centered at v using L_p normalization $\|v^* - v\|_p$, where p can be 1, 2 or ∞. Here in this paper, we mainly focus on the L_∞ norm and SSIM similarity (Wang et al., 2004) for comparison to clean frames.

Although there are multiple evaluation metrics for SOT across various challenges, we decide to explore two standards that are in most common use for visual tracking,

represented as $\mathcal{A}$ and $\mathcal{R}$, short for accuracy and robustness. $\mathcal{A}$ denotes the average of *IoU* scores of all frames that contain overlapping perturbed bounding boxes and predicted bounding boxes until the end of video or reinitialization. $\mathcal{R}$ weights the tracking performance according to the number of failed frames in a discounted reward manner. These two values can be calculated as:

$$IoU_i = \frac{\hat{B}_i \cap B_i}{\hat{B}_i \cup B_i}, \quad ro_i = \begin{cases} 1, & IoU_i \in (0,1], \\ 0, & else. \end{cases} \tag{1}$$

$$\mathcal{A} = \frac{1}{N} \times \sum_{i}^{N} (\gamma_a)^{i//L} IoU_i * ro_i, \quad \mathcal{R} = \sum_{i}^{N} (\gamma_r)^{i//L} ro_i \tag{2}$$

where IoU_i represents *Intersection over Union* between predicted $\hat{B}_i$ and ground truth $\hat{B}_i$. γ_a and γ_r state discounted factors for accuracy and robustness, highlighting the impact of subsequent tracking performance across the video clip. Similar to SPARK (Guo et al., 2020), we split the video into several intervals with length L based on Frame Per Second.

Algorithm 1 Momentum-based Perturbation Generation in OPE

Input: SOT tracker $\mathcal{T}$, clean video v, adversarial video $v^* = v$, maximum perturbation ϵ, candidate number C, momentum factor μ, trade-off factor ι, iterations k, initial gradient g_0, tracking performance $\mathcal{TP} = 1$, adversarial candidate set $\mathcal{V}$.

Output: adversarial candidate set $\mathcal{V}$

$\mathcal{A}, \mathcal{R} = \mathcal{T}(v;\ \theta)$
while $\|v_0^* - v_0\|_\infty \leq \epsilon$ **do**
  for $i = 0$ to $C - 1$ **do**
    $adv = \mathcal{N}(0, I;\ v_0.shape); \quad g' = \frac{adv}{\|adv\|_\infty}$
    $\mathcal{A}^*, \mathcal{R}^* = \mathcal{T}(v_0^* + \frac{\epsilon}{k} \times Sign(\mu \times g_0 + g');\ \theta)$
    if $\iota \times \frac{\mathcal{A}^*}{\mathcal{A}} + (1 - \iota) \times \frac{(\mathcal{R}+1)}{(\mathcal{R}^*+1)} < \mathcal{TP}$ **then**
      $g_{opt} = \mu \times g_0 + g'$; $\mathcal{TP} = \iota \times \frac{\mathcal{A}^*}{\mathcal{A}} + (1 - \iota) \times \frac{(\mathcal{R}+1)}{(\mathcal{R}^*+1)}$
    end if
  end for
  $v_0^* = v_0^* + \frac{\epsilon}{k} \times Sign(g_{opt}); \quad g_0 = g_{opt}; \quad \mathcal{V}.append(v_0^*)$
end while
Return $\mathcal{V}$

Generally, attacks on SOT can be categorized into untargeted and targeted attacks. In this paper, we mainly focus on untargeted attacks, generating adversarial videos based on object motions to degrade the overall tracking performance or deviate the tracker across the whole video clip.

3.2 Heavy perturbation generator

In the first stage of our algorithms, we generate a group of heavily perturbed videos as candidate adversarial examples. To diversify perturbations and increase the probability of

successful attacks, we synergistically exploit patch-based and momentum-based perturbation generators.

In the *patch-based* perturbation generating process, we randomly select a certain number of candidate videos from the dataset consisting of the attacked video. For each candidate video, we randomly pick up a frame and crop an image patch using a window that has the same size as the ground truth bounding box in the initial frame of the attacked video. Then we can construct a set of cropped patches that can be added to the initial frame of the target video. Different from classification tasks such as video recognition or human action recognition, where we regard each video frame as a whole to feed the underlying model and extract feature representation, the final objective of all SOT problems is to accurately locate objects in subsequent frames, therefore it is intuitively to craft noises on the region surrounding the object instead of marginal regions. To do so, we randomly select a group of areas from the search region around the initial frame, and craft patch perturbations over these areas, then we are capable of adding these perturbations to the adversarial candidate set $\mathcal{V}$.

On the other hand, we propose a *momentum-based* perturbation generator, which estimates gradient directions by accumulating historical velocity vectors. IoU Attack (Jia et al., 2021) leverages this concept and extends it to the temporal correspondence among continuous frames. Inspired by this, we delve into the spatial correspondence following MI-FGSM. As illustrated in Algorithm 1, after collecting patch-based adversarial candidates in the set $\mathcal{V}$, for each perturbing level $\frac{\epsilon}{k}$, where ϵ is the overall adversarial magnitude, and k is the number of iterations, we randomly sample C perturbing directions denoted as $g^{'}$, then adversaries are crafted along the historically optimal direction progressively until the magnitude of perturbation exceeds the ϵ-ball bound around the initial frame v_0. Balanced by trade-off factor ι, if the tracking performance decreases, we then update and get the optimal gradient g_{opt} with momentum. With the momentum-based generator, we can get optimal adversarial frames in each perturbing level. Particularly, if any of these adversaries provides better attack results than previous patch-based perturbations, we can directly output this adversary as shown in Fig. 2. Cases with reinitialization (VOT2018) can be easily extended by repeating the previous process on all reinitialized frames step by step.

3.3 Key patch selection

As illustrated above, some areas in initialized frames are more beneficial for feature representations of the target object, but others are not. Take video *Bird1* in Fig. 2 for instance, perturbations added on edges of bounding boxes affect the tracking performance much less than those on object-surrounding ones. Therefore, removing perturbations attached to those regions will not affect the overall attack results but increase the similarities between original frames and perturbed ones. As shown in Fig. 2, we impose a mask that is split into $\mathcal{P} \times \mathcal{P}$ patches and element-wisely composed of all 1s. Considering computational efficiency as well as the averaged size of video frames across different datasets, we adjust $\mathcal{P}$ as a hyper-parameter and conduct a grid search. Then we apply a reinforcement learning (RL)-based key patch selection framework, which is implemented by

Actor-Critic network $\mathcal{Z}$, to select the least important patch step by step until the RL agent enters into a terminal state.

As shown in the second part of Fig. 2, our network contains 5 convolutional layers, each of them is followed by a max-pooling layer, where parameters are shared between Actor and Critic branches, and extract features of newly added perturbations. However, the shape of videos can be varied even in the same tracking dataset. Resizing them into a fixed size may result

in unwanted geometric distortion, which is extremely harmful to localizing objects in SOT. Therefore we introduce a Spatial Pyramid Pooling (SPP) (He et al., 2016) strategy on top of the last convolutional layer to remove the fixed size constraint of the network. Subsequently, we append 3 fully connected layers to estimate what is the best action that the agent should take and the corresponding critic value of that.

Generally, we consider the key patch selection as a multi-step Markov Decision Process (MDP), which contains states, actions, transition function, and a reward function. In our task, the state s_t at time step t is defined as the pixel-wise difference between v_0 and v_0^* masked by the current mask $M_t \in \mathbb{R}^{S\times\mathcal{P}\times\mathcal{P}}$. It can be denoted as:

$$s_t = (v_0^* - v_0) \odot M_t \tag{3}$$

where $\odot$ represents Hadamard product. At time step 0, M_0 is $\{1\}^{S\times\mathcal{P}\times\mathcal{P}}$. An action $a_t = \mathcal{Z}(s_t)$ refers to a $S \times \mathcal{P}^2$ softmax matrix, indicating the least important patch in each initialized frame to successfully track the target at time step t. Then once the agent chooses an action a_t, we can set the corresponding element in M_t to 0.

Algorithm 2 Key Patch Selection and Sign Attack Module in OPE

Input: SOT tracker $\mathcal{T}$, clean video clip v, pretrained policy θ_p, value network parameter θ_c, adversarial candidate set $\mathcal{V}$, video candidate number n, gradient candidate number K, smoothing parameter ρ_d, and direction learning step size α. number of attack queries $\mathcal{N}_\mathcal{A}$, initial grid mask M

Output: adversarial example set $\mathcal{V}$

Fine-tune Actor-Critic network parameters θ_p and θ_c using top-n patch-based candidates from $\mathcal{V}$ that is Ranked based on $\mathcal{TP}$ in Algorithm 1 and $\|\mathcal{V}_i - v_0\|_\infty$ with ascending order.
for $i = 0$ to n **do**
  $\mathcal{A}, \mathcal{R} = \mathcal{T}(\mathcal{V}_i;\ \theta)$
  Apply policy θ_p to get sparse mask M_i
  $\mathcal{A}^*, \mathcal{R}^* = \mathcal{T}((\mathcal{V}_i - v_0) \odot M_i + v_0;\ \theta)$
  if $\gamma\frac{\mathcal{A}^*}{\mathcal{A}} + (1-\gamma)\frac{\mathcal{R}}{\mathcal{R}^*} \leq \frac{\gamma(\tau_1\tau_2-1)+1}{\tau_2}$ **then**
    $\phi_d = \frac{(\mathcal{V}_i - v_0)}{\|\mathcal{V}_i - v_0\|_\infty}$; Using binary search algorithm to compute $g(\phi_d)$
  end if
  for $n_A = 0$ to $\mathcal{N}_\mathcal{A}$ **do**
    Randomly sample K vectors $u_1, ..., u_k$ using $\mathcal{N}(0, I)$
    $\hat{\nabla} g(\phi_d) = \frac{1}{K}\sum_{k=1}^{K} Sign(g(\phi_d + \rho_d u_k) - g(\phi_d))u_k$
    $\phi_d = \phi_d - \alpha\hat{\nabla} g(\phi_d)$
    Recompute $g(\theta_d)$ as shown above.
  end for
  $\mathcal{V}_i = v_0 + g(\phi_d)\phi_d$
end for
Ranking $\mathcal{V}$
Return $\mathcal{V}$

Denoting this process as a function $\mathcal{F}$, we can update the state to

$$s_{t+1} = (v_0^* - v_0) \odot \mathcal{F}(M_t, a_t) \tag{4}$$

s_{t+1} will be the terminal state if $a_t \in \{a_0, a_1, \ldots, a_{t-1}\}$ or $\frac{\mathcal{A}(\mathcal{T}(v_0+s_{t+1}))}{\mathcal{A}(\mathcal{T}(v_0+s_0))} > \tau_1$ or $\frac{\mathcal{R}(\mathcal{T}(v_0+s_{t+1}))}{\mathcal{R}(\mathcal{T}(v_0+s_0))} < \tau_2$. Since SOT is inherently a regression problem within the continuous output space instead of a pure classification problem, slight manipulation of the adversarial perturbation may be reflected in the final tracking results. Therefore we introduce ratio thresholds τ_1 and τ_2 to maintain the attack results within an acceptable scale. Generally, our goal is to delete less important patches and maximize the long-term expected reward, therefore we design the reward in step t as

$$r_t = \begin{cases} 0, & a_t \in \{a_0, a_1, .., a_{t-1}\}; \\ -1, & \frac{\mathcal{A}(\mathcal{T}(v+s_{t+1}))}{\mathcal{A}(\mathcal{T}(v+s_t))} > \tau_1 \ or \ \frac{\mathcal{R}(\mathcal{T}(v+s_{t+1}))}{\mathcal{R}(\mathcal{T}(v+s_t))} < \tau_2; \\ \gamma \frac{\mathcal{A}(\mathcal{T}(v+s_t))}{\mathcal{A}(\mathcal{T}(v+s_{t+1}))} + (1-\gamma)\frac{\mathcal{R}(\mathcal{T}(v+s_{t+1}))}{\mathcal{R}(\mathcal{T}(v+s_t))}, & else \end{cases}$$

In the offline training stage, we select a certain number of candidate videos generated from the previous step, then feed them into policy network $\pi_{\theta_c}(a_t \| s_t)$ and critic network $\pi_{\theta_c}(c_t \| s_t)$ to maximize the expected long-term reward with PPO algorithm, which is written as

$$L(\theta_p) = \sum_{(s_t, a_t)} \min \left(\frac{\pi_{\theta_p}(a_t \| s_t)}{\pi_{\theta_p^{old}}(a_t \| s_t)}, clip\left(\frac{\pi_{\theta_p}(a_t \| s_t)}{\pi_{\theta_p^{old}}(a_t \| s_t)}, 1-\rho, 1+\rho \right) \right) A_{\theta_p^{old}}(s_t \| a_t) \tag{5}$$

where $A_{\theta_p}(s_t \| a_t) = Q_{\theta_p}(s_t, a_t) - V_{\theta_c}(s_t) = \gamma^{T-t} V(s_T) + \gamma^{T-t-1} r_{T-1} + \cdots + r_t - V_{\theta_c}(s_t)$, Q_{θ_p} is the Q-value calculated by discounting future rewards, V_{θ_c} is the critic value generated by critic network. ρ denotes the clip parameter to regularize policy iterations.

3.4 Sign attack module

As indicated in Algorithm 2, after removing less important patch-level perturbations attached to initial frames of videos, we can fetch manipulated adversarial examples as well as their tracking accuracy and robustness. Then we need a boundary walking method to help us compress the noise magnitude while maintaining attack results within a specific scope. As shown in part (c) of Fig. 2, we iteratively update victim frame v_0 until its magnitude is compressed from ϵ_1 to ϵ_3, while maintaining competitive attack results or even strengthening it. Cheng et al. (2018) states that a black-box attack problem can be formulated into an optimization problem, where the objective function can be evaluated as a binary search with additional model queries. Then a zeroth-order optimization algorithm can be applied to solve this optimization problem. In this paper, we exploit the Sign-OPT algorithm in the Sign Attack Module.

In our approach, ϕ_d and $g(\phi_d)$ indicate our designated search direction and corresponding distance from the initial frame v_0 to its nearest adversarial example that has the same or similar tracking results within a predefined threshold along ϕ_d. The objective function can be written as

$$\min_{\phi_d} g(\phi_d), \quad where \ g(\phi_d) = \arg\min_{\lambda} \left(\mathcal{AR}\left(\mathcal{T}\left(v_0 + \lambda \frac{\phi_d}{\|\phi_d\|}; \theta \right) \right) \leq \frac{\gamma(\tau_1\tau_2 - 1) + 1}{\tau_2} \right) \tag{6}$$

where τ_1 and τ_2 are hyper-parameters exploited in Key Patch Selection. As the evaluation results of SOT, $\mathcal{AR}$ is denoted as $\gamma \frac{\mathcal{A}(\mathcal{T}(v_0+\lambda \frac{\phi_d}{\|\phi_d\|}))}{\mathcal{A}(\mathcal{T}(v_0+s_0))} + (1-\gamma)\frac{\mathcal{R}(\mathcal{T}(v_0+s_0))}{\mathcal{R}(\mathcal{T}(v_0+\lambda \frac{\phi_d}{\|\phi_d\|}))}$. We need to estimate its directional derivative by consuming a huge amount of queries when computing $g(\phi_d + u) - g(\phi_d)$. However, it will take a large number of computational resources if we intend to obtain the gradient derivative accurately. Due to the various and large dimensions of our input, we decide to improve query complexity by an imperfect but informative estimation of directional derivative. Therefore, we exploit the sign value and compute the gradient by sampling K gaussian vectors:

$$\hat{\nabla} g(\phi_d) = \frac{1}{K}\sum_{k=1}^{K} Sign(g(\phi_d + \rho_d u_k) - g(\phi_d))u_k \tag{7}$$

When starting an attack on videos, we need to initialize perturbing directions $\phi_d = \frac{v_0^* - v_0}{\|v_0^* - v_0\|}$, where v_0^* can be retrieved by sampling from v_0's candidate adversarial sets $\mathcal{V}$, including patch-based and momentum-based perturbations. Detailed in Algorithm 2, by trading off the magnitude of adversaries and their tracking performance, we rank the candidate list with $\mathcal{TP}$ and L_1 normalization and pick the top-n target video clips for the attacked video.

4 Experiments

In this section, we describe our experimental settings and analyze the effectiveness of the proposed DIMBA algorithm against different trackers on four challenging short-term or long-term datasets, including OTB100 (Wu et al., 2015), VOT2018 (Kristan, 2018), UAV123 (Mueller et al., 2016), and LaSOT (Fan et al., 2019). Part of the qualitative tracking results performed by SiamRPN++ is shown in Fig. 3.

4.1 Experimental settings

Victim models As mentioned in Sect. 1, current tracking models can be divided into Siamese-based, discrimination, and reinforcement learning-based trackers. Considering overall tracking performance, we select one or more most representative trackers for each of them, which consists of SiamRPN++ that uses MobileNetv2 (Sandler et al., 2018), and ResNet50 (He et al., 2016) as backbones, DaSiamRPN (Zhu et al., 2018), PrDiMP50 (Danelljan et al., 2020), TrTr (Zhao et al., 2021), and Action-Decision Network (Yun et al., 2017). Specifically, SiamRPN++(R) exploits ResNet50 as the backbone model, while SiamRPN++(M) utilizes MobileNetv2.

Metrics To fairly compare our attack results with original tracking performance and previous black-box attacks on SOT, standard evaluation methods are exploited. While testing DIMBA on OTB100 (Wu et al., 2015), UAV123 (Mueller et al., 2016) and LaSOT (Fan et al., 2019), we utilize precision and success rate in a one-pass evaluation (OPE) scenario. As for the VOT2018 challenge (Kristan, 2018), we introduce a reinitialization mechanism five frames after the tracker lost the target.

Computing infrastructures We conduct experiments on a computer with three Nvidia GeForce RTX 2080Ti and one Nvidia GeForce RTX 3090 GPUs, an Intel(R) Core(TM) i9-10900X CPU @ 3.70 GHz, running Ubuntu 18.04.5 LTS.

Fig. 3 Illustration of clean and adversarial tracking results tracked by SiamRPN++ tested on OTB100. Green bounding boxes indicate ground truth locations, blue ones state originally predicted locations, while red ones demonstrate adversarially attacked locations (Color figure online)

4.2 Implementation details

Our experiment is implemented in PyTorch.[1] In momentum-based perturbation generation, maximum noise magnitude ϵ is 8 (following One-Shot imperceptible settings) , candidate number C is 25, iteration number k is 128, momentum factor μ is 0.5, trade-off factor ι is 0.4. Same to momentum generator, the patch-based generator produces adversarial sets with capacity C as well. γ_a and γ_r are both set to be 0.9.

To pre-train the Actor-Critic Network for key patch selection, we set PPO epoch, clipping parameter ρ, buffer capacity, and maximum gradient normalization to 10, 0.2, 500, and 0.5, respectively. As for patch number $\mathcal{P}$, we exploit the grid search strategy and set

[1] We release our code via GitHub https://github.com/TrustAI/DIMBA.

$\mathcal{P}$ as 2, 4, 8, 16, 32. For balancing selection efficiency and final impact on tracking performance, $\mathcal{P}$ is parameterized to 16.

In the same way, the combination of ratio threshold τ_1 and τ_2 is set to 1.5 and 0.4. trade-off factor γ is set to 0.5, video candidate number n is naturally set to 20 out of 30, gradient candidate number K is assigned to be 100, and the number of attack queries $\mathcal{N}_{\mathcal{A}}$ can be 60.

4.3 Overall attack results

Results on VOT2018 Table 2 compares the overall results of these trackers on the VOT2018 dataset. We exploit randomly generated noises as well as perturbations computed by IoU Attack (Jia et al., 2021) and compare them with our proposed method. Specifically, our algorithm outperforms IoU Attack concerning accuracy in DaSiamRPN and ADNet by 8.45 and 5.82%, respectively. Furthermore, in terms of robustness, our approach exceeds IoU Attack in SiamRPN++(ResNet50), DaSiamRPN, and ADNet by 9.32, 3.21, and 2.97%. As for EAO (Expected Average Overlap) in SiamRPN++ and ADNet, we have achieved 6.2 and 7.9% improvement.

Results on OTB100 As shown in Fig. 4, we draw success and precision plots of various trackers selected according to their categories and tested on OTB100. Compared to the

Table 2 Attack results of SiamRPN++ (Li et al., 2019), DaSiamRPN (Zhu et al., 2018), PrDiMP50 (Danelljan et al., 2020), ADNet (Yun et al., 2017), and TrTr (Zhao et al., 2021) on VOT2018 (Kristan, 2018), evaluated using Accuracy, Robustness, and EAO (expected average overlap)

Trackers	Accuracy↑			
	Original (%)	Random (%)	IoU Attack (%)	Ours (%)
SiamRPN++(R)	60.30	59.12	**56.84**	57.01
DaSiamRPN	58.52	57.14	53.19	**48.68**
PrDiMP50	61.80	60.86	**57.29**	58.12
ADNet	50.80	48.28	39.53	**37.14**
TrTr	60.65	60.12	**57.88**	58.84
Trackers	Robustness↓			
	Original	Random	IoU attack	Ours
SiamRPN++(R)	0.235	0.289	1.169	**1.278**
DaSiamRPN	0.276	0.295	1.214	**1.253**
PrDiMP50	0.165	0.171	**0.377**	0.352
ADNet	0.314	0.337	1.412	**1.454**
TrTr	0.110	0.121	**0.227**	0.193
Trackers	EAO (expected average overlap)↑			
	Original	Random	IoU Attack	Ours
SiamRPN++(R)	0.415	0.351	0.129	**0.121**
DaSiamRPN	0.382	0.347	**0.124**	0.159
PrDiMP50	0.442	0.425	**0.275**	0.311
ADNet	0.329	0.317	0.113	**0.104**
TrTr	0.493	0.488	**0.336**	0.343

Bold indicates the best performance among all black-box attacks (including Random, IoU Attack, and Ours)

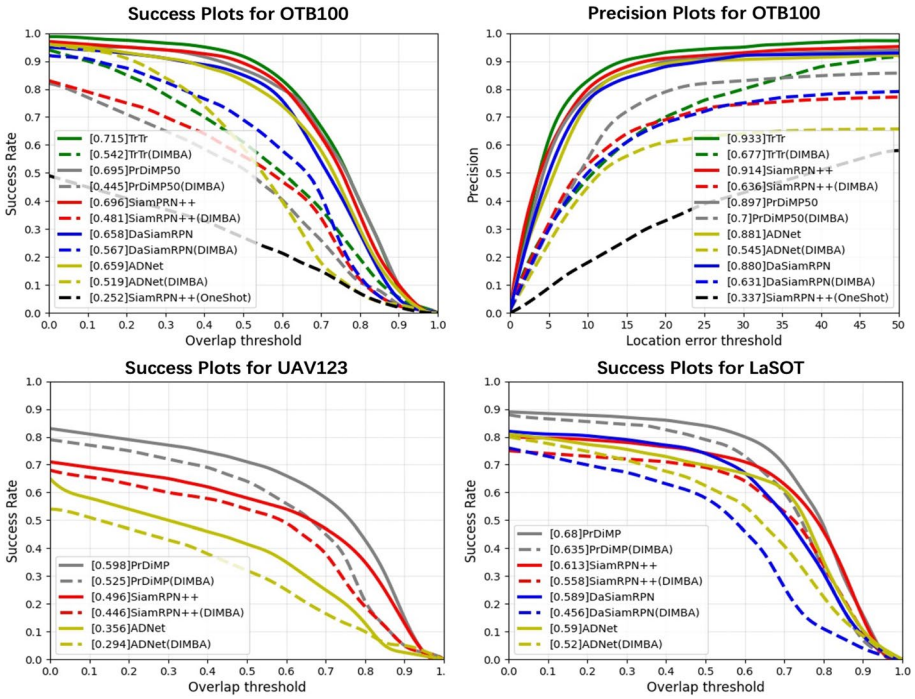

Fig. 4 Illustration of success plots and precision plots tested on OTB100, UAV123, and LaSOT. Success plots represent the AUC values regarding different overlapping scores, while precision plots are at the error threshold of 20 pixels with respect to centered location errors. The numbers in brackets in front of tracker names denote AUC scores

original tracking performance, our black-box attack method can reduce the AUC score and visually change the curves' shape. Meanwhile, we correspondingly visualize the results of a white-box One-Shot Attack (Chen et al., 2020) and check the difference. Meanwhile, Table 3 illustrates the success and precision rates of original videos, random perturbations, One-Shot Attack, IoU Attack, and our method.

Results on UAV123 and LaSOT Depicted in Fig. 4, tracking results of different trackers are illustrated based on UAV123 and LaSOT. With our attack method, the AUC score of success plots tested on UAV123 are decreased by 4.3, 10.8, and 17.4% for PrDiMP50, SiamRPN++(ResNet50), and ADNet individually. In the meantime, the same score of success plots calculated on LaSOT are reduced by 6.6, 9.0, 22.5, and 11.8% for PrDiMP50, SiamRPN++, DaSiamRPN, and ADNet respectively.

4.4 Ablation study of key patch selection

We conduct a series of experiments to evaluate the impact of the key patch selection module. SiamRPN++(R), DaSiamRPN, and ADNet are selected as our baselines, and tracking results on OTB100 and UAV123 are shown in Fig. 5. We query fewer times in black-box settings to reach a similar perturbation magnitude ϵ using Key Patch Selection. Meanwhile, the average IoU scores in subintervals as shown in Fig. 5 under our proposed Key Patch

Table 3 Attack Results of SiamRPN++(ResNet50), SiamRPN++(Mobilev2), DaSiamRPN, ADNet, TrTr, and PrDiMP50 on OTB100 (Wu et al., 2015), evaluated using success rate and precision

Trackers	Success rate↑				
	Original (%)	Random (%)	IoU Attack (%)	One-Shot Attack (%)	Ours (%)
SiamRPN++(R)	69.64	65.21	49.58	*25.22*	**48.09**
SiamRPN++(M)	66.06	59.41	**42.73**	*35.94*	45.02
DaSiamRPN	65.82	63.91	**53.24**	*37.60*	56.66
ADNet	63.71	61.76	53.80	*30.98*	**51.92**
TrTr	71.53	68.32	56.32	*41.88*	**54.16**
PrDiMP50	69.50	66.03	46.54	*28.10*	**44.52**
Trackers	Precision↑				
	Original (%)	Random (%)	IoU Attack (%)	One-Shot Attack (%)	Ours (%)
SiamRPN++(R)	91.42	86.13	**63.19**	*33.68*	63.68
SiamRPN++(M)	86.43	79.76	62.18	*26.41*	**61.29**
DaSiamRPN	86.50	81.23	64.78	*29.65*	**63.09**
ADNet	88.13	84.15	**51.20**	*20.85*	54.55
TrTr	92.81	87.86	68.74	*45.85*	**67.66**
PrDiMP50	89.73	87.24	70.88	*38.10*	**69.96**

As OPE (one pass evaluation) dataset, OTB100 can also be perturbed by white-box attacks, like One-Shot Attack (Chen et al., 2020), which as it should be, outperforms black-box algorithms, and is highlighted in italic font

Bold indicates the best performance among all black-box attacks (including Random, IoU Attack, and Ours)

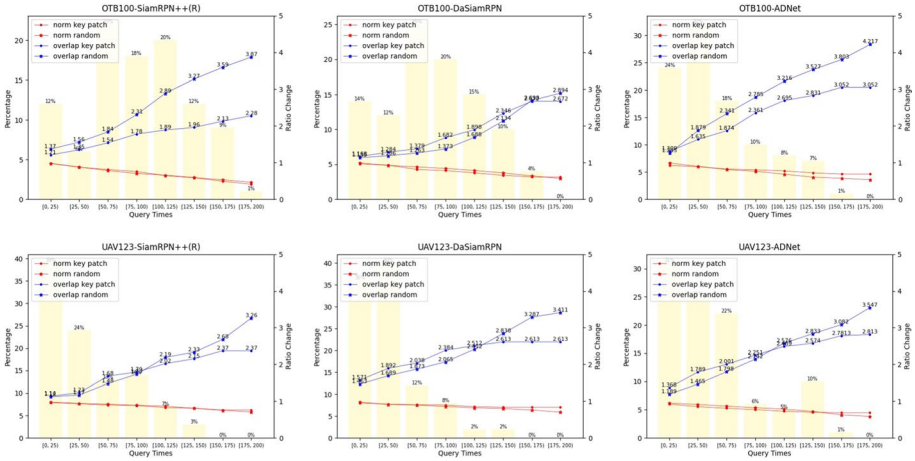

Fig. 5 Illustration of the ablation study on key patch selection module of our proposed DIMBA Attack. Results are conducted over OTB100 UAV123, tracked by SiamRPN++(ResNet), DaSiamRPN, ADNet. Yellow bars indicate the percentage of query times in 8 subintervals from 0–200. Red lines represent ratio changes in l_∞-norm adversarial magnitude, while blue lines state changes of average overlap scores in each interval. — ● — and — ● — state changes under our proposed key patch selection algorithm. In contrast, — ∗ — and — ∗ — refer to ones by randomly selected (Color figure online)

Table 4 Comparison of average query times between IoU Attack and our proposed method, tracked by SiamRPN++(R), DaSiamRPN, PrDiMP50, and ADNet, tested on OTB100, UAV123, and LaSOT

Datasets	Query Times(IoU‖Ours)↓			
	SiamRPN++(R)	DaSiamRPN	PrDiMP50	ADNet
OTB100	81460‖**43295**	79365‖**42981**	82727‖**41026**	76352‖**35284**
UAV123	129802‖**108901**	98158‖**52830**	77102‖**56138**	48223‖**26815**
LaSOT	228570‖**186285**	200382‖**129483**	182939‖**115024**	93577‖**62661**

Bold indicates the best performance among all black-box attacks (including Random, IoU Attack, and Ours)

Table 5 Average SSIM similarity between clean videos and perturbed videos from OTB100, UAV123, and LaSOT, tracked by SiamRPN++(R), PrDiMP50, and ADNet

Datasets	SSIM(One-Shot‖IoU‖Ours)↑		
	SiamRPN++(R)	PrDiMP50	ADNet
OTB100	*0.9958*‖0.9905‖**0.9937**	*0.9947*‖0.9921‖**0.9932**	*0.9984*‖0.9937‖**0.9963**
UAV123	*0.9973*‖0.9856‖**0.9913**	*0.9939*‖0.9914‖**0.9918**	*0.9981*‖0.9948‖**0.9952**
LaSOT	*0.9992*‖0.9815‖**0.9862**	*0.9996*‖0.9945‖**0.9957**	*0.9998*‖**0.9986**‖0.9978

Bold indicates the best performance among all black-box attacks (including Random, IoU Attack, and Ours) and italic indicates the performance of the white-box attack (i.e., One-Shot Attack)

Selection algorithm majorly remain smaller than the ones with random patch selection or without the Key Patch Selection module.

4.5 Comparison with previous works

According to our understanding, the overall computational complexity of IoU Attack (Jia et al., 2021) is $\mathcal{O}(KNL)$, where K is the number of epochs for choosing perturbations on each frame, N is the candidate number of random noises, L is the length of the video clip. Whereas in our algorithm, our query complexity can be reduced to $\mathcal{O}(KN + C)$, where C is a constant number independent of L. The comparison in query efficiency between query-based black box attack algorithms, IoU Attack, and our proposed method, is illustrated in Table 4. In the meantime, we compare the SSIM similarity between clean and adversarial videos to qualitatively verify the side effects of our proposed algorithm, the result is shown in Table 5. Except for some specific cases, our algorithm achieves better SSIM similarity than the query-based IoU Attack.

5 Conclusions

In this work, we propose an effective and efficient query-based black-box attack for SOT. An Actor-Critic key patch selection module is exploited to reduce redundant noises and increase query efficiency. Meanwhile, the combination of patch-based and

momentum-based perturbation generators diverse potential adversarial directions and introduce heavily damaged tracking performance. Compared with existing works, our method requires fewer queries on SOT and less perturbation from the perspective of a whole video clip but maintains competitive, even better manipulating results. The experiments in both long-term and short-term datasets across three major categories of trackers demonstrate the effectiveness of our framework. We hope this work can elucidate the source of vulnerabilities in these trackers, optimistically paving the way for more powerful ones.

Author contributions XY contributed to the idea, algorithm, theoretical analysis, writing, and experiments. WR contributed to the idea, theoretical analysis, and writing. JF contributed to the writing.

Funding This work is supported by Partnership Resource Fund on EPSRC project on ORCA Hub [EP/R026173/1].

Data availability Our code is available on https://github.com/TrustAI/DIMBA.

Declarations

Conflict of interest The authors declare that they have no conflict of interest

References

Bertinetto, L., Valmadre, J., Henriques, J. F., Vedaldi, A., & Torr, P. H. (2016). Fully-convolutional siamese networks for object tracking. arXiv:1606.09549

Chen, X., Yan, X., Zheng, F., Jiang, Y., Xia, S. T., Zhao, Y., & Ji, R. (2020). One-shot adversarial attacks on visual tracking with dual attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10176–10185).

Cheng, M., Le, T., Chen, P.Y., Yi, J., Zhang, H., & Hsieh, C. J. (2018). Query-efficient hard-label black-box attack: An optimization-based approach. arXiv preprint arXiv:1807.04457

Danelljan, M., Bhat, G., Khan, F. S., & Felsberg, M. (2019). ATOM: Accurate tracking by overlap maximization. arXiv:1811.07628

Danelljan, M., Gool, L. V., & Timofte, R. (2020). Probabilistic regression for visual tracking. arXiv:2003.12565

Ding, L., Wang, Y., Yuan, K., Jiang, M., Wang, P., Huang, H., & Wang, Z. J. (2021). Towards universal physical attacks on single object tracking. In *AAAI* (pp. 1236–1245).

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T. & Song, D. (2018). Robust physical-world attacks on deep learning models. arXiv:1707.08945

Fan, H., Lin, L., Yang, F., Chu, P., Deng, G., Yu, S., Bai, H., Xu, Y., Liao, C., & Ling, H. (2019). LaSOT: A high-quality benchmark for large-scale single object tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 5374–5383).

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015). Explaining and harnessing adversarial examples. arXiv:1412.6572

Grosse, K., Papernot, N., Manoharan, P., Backes, M., & McDaniel, P. (2017). Adversarial examples for malware detection. In *European symposium on research in computer security* (pp. 62–79). Springer.

Guo, Q., Xie, X., Juefei-Xu, F., Ma, L., Li, Z., Xue, W., Feng, W., & Liu, Y. (2020). Spark: Spatial-aware online incremental attack against visual tracking. arXiv:1910.08681

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*.

Huang, L., Gao, C., Zhou, Y., Xie, C., Yuille, A. L., Zou, C., & Liu, N. (2020). Universal physical camouflage attacks on object detectors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 720–729).

Jia, S., Song, Y., Ma, C., & Yang, X. (2021). Iou attack: Towards temporally coherent black-box adversarial attack for visual object tracking. arXiv:2103.14938

Jia, Y., Lu, Y., Shen, J., Chen, Q. A., Zhong, Z., & Wei, T. (2019). Fooling detection alone is not enough: First adversarial attack against multiple object tracking. arXiv preprint arXiv:1905.11026

Kristan, M. (2018). The sixth visual object tracking vot2018 challenge results. In *Proceedings of the European conference on computer vision (ECCV) workshops*.

Kurakin, A., Goodfellow, I., & Bengio, S. (2017). Adversarial examples in the physical world. arXiv:1607.02533

Li, B., Wu, W., Wang, Q., Zhang, F., Xing, J., & Yan, J. (2019). Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 4282–4291).

Li, B., Yan, J., Wu, W., Zhu, Z., & Hu, X. (2018). High performance visual tracking with siamese region proposal network. In *2018 IEEE/CVF conference on computer vision and pattern recognition* (pp. 8971–8980). https://doi.org/10.1109/CVPR.2018.00935

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2019). Towards deep learning models resistant to adversarial attacks. arXiv:1706.06083

Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). Human-level control through deep reinforcement learning. *Nature, 518*, 529–533.

Moosavi-Dezfooli, S. M., Fawzi, A., & Frossard, P. (2016). Deepfool: A simple and accurate method to fool deep neural networks. arXiv:1511.04599

Mueller, M., Smith, N., & Ghanem, B. (2016). A benchmark and simulator for UAV tracking. In *European conference on computer vision* (pp. 445–461). Springer.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. arXiv:1506.02640

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4510–4520).

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., & Dieleman, S. (2016). Mastering the game of go with deep neural networks and tree search. *Nature, 529*(7587), 484–489.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2014). Intriguing properties of neural networks. arXiv:1312.6199

Wang, Z., Bovik, A., Sheikh, H., & Simoncelli, E. P. (2004). Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing, 13*(4), 600–612. https://doi.org/10.1109/TIP.2003.819861

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine Learning, 8*(3), 279–292.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning, 8*(3), 229–256.

Wu, Y., Lim, J., & Yang, M. H. (2015). Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 37*(9), 1834–1848. https://doi.org/10.1109/TPAMI.2014.2388226

Xie, C., Wang, J., Zhang, Z., Zhou, Y., Xie, L., & Yuille, A. (2017). Adversarial examples for semantic segmentation and object detection. arXiv:1703.08603

Yakura, H., & Sakuma, J. (2019). Robust audio adversarial example for a physical attack. In *Proceedings of the twenty-eighth international joint conference on artificial intelligence*. https://doi.org/10.24963/ijcai.2019/741

Yan, B., Wang, D., Lu, H., & Yang, X. (2020a). Cooling-shrinking attack: Blinding the tracker with imperceptible noises. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 990–999)

Yan, X., Chen, X., Jiang, Y., Xia, S. T., Zhao, Y., & Zheng, F. (2020b). Hijacking tracker: A powerful adversarial attack on visual tracking. In *ICASSP 2020—2020 IEEE international conference on acoustics, speech and signal processing (ICASSP)* (pp. 2897–2901). IEEE.

Yun, S., Choi, J., Yoo, Y., Yun, K., & Young Choi, J. (2017). Action-decision networks for visual tracking with deep reinforcement learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2711–2720).

Zhang, H., Zhou, H., Miao, N., & Li, L. (2019a). Generating fluent adversarial examples for natural languages. In *Proceedings of the 57th annual meeting of the association for computational linguistics. association for computational linguistics, Florence, Italy* (pp. 5564–5569). https://doi.org/10.18653/v1/P19-1559

Zhang, L., Gonzalez-Garcia, A., van de Weijer, J., Danelljan, M., & Khan, F. S. (2019b). Learning the model update for siamese trackers. arXiv:1908.00855

Zhao, M., Okada, K., & Inaba, M. (2021). TrTr: Visual tracking with transformer. arXiv:2105.03817

Zhu, Z., Wang, Q., Li, B., Wu, W., Yan, J., & Hu, W. (2018). Distractor-aware siamese networks for visual object tracking. arXiv:1808.06048