

Temporally-Transferable Perturbations: Efficient, One-Shot Adversarial Attacks for Online Visual Object Trackers

Krishna Kanth Nakka¹ and Mathieu Salzmann^{1,2}

¹ EPFL Computer Vision Lab, ² ClearSpace SA

Abstract

In recent years, the trackers based on Siamese networks have emerged as highly effective and efficient for visual object tracking (VOT). While these methods were shown to be vulnerable to adversarial attacks, as most deep networks for visual recognition tasks, the existing attacks for VOT trackers all require perturbing the search region of every input frame to be effective, which comes at a non-negligible cost, considering that VOT is a real-time task. In this paper, we propose a framework to generate a single temporally-transferable adversarial perturbation from the object template image only. This perturbation can then be added to every search image, which comes at virtually no cost, and still successfully fool the tracker. Our experiments evidence that our approach outperforms the state-of-the-art attacks on the standard VOT benchmarks in the untargeted scenario. Furthermore, we show that our formalism naturally extends to targeted attacks that force the tracker to follow any given trajectory by precomputing diverse directional perturbations.

1. Introduction

Visual Object Tracking (VOT) [23] is a key component of many vision-based systems, such as surveillance and autonomous driving ones. Studying the robustness of object trackers is therefore critical from a safety and security point of view. When using deep learning, as most modern tackers do, one particular security criterion is the robustness of the deep network to adversarial attacks, that is, small perturbations aiming to fool the prediction of the model. In recent years, the study of such adversarial attacks has become an increasingly popular topic, extending from image classification [11, 22] to more challenging tasks, such as object detection [43] and segmentation [1, 10].

VOT is no exception to this rule. However, unlike in recognition-driven tasks, where the predictions are typically restricted to a predefined label set, such as image categories in ImageNet or object categories in PASCAL VOC,

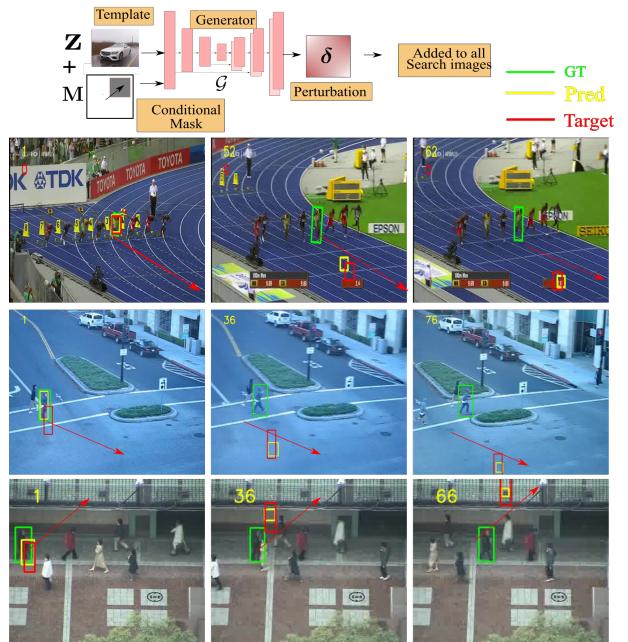


Figure 1. Our approach generates a single transferable perturbation from the object template in 8ms. We can then use this perturbation to attack the tracker throughout the *entire* sequence, either in an untargeted manner, or a targeted one by forcing it to follow a predefined motion, such as a fixed direction as illustrated above, or a more complicated trajectory, as shown in our experiments.

the category of the objects tracked at test time by VOT methods is not known beforehand and only defined by the first frame of the input video. In other words, adversarial attacks on VOT systems should generalize to object templates that were not seen during the training phase. Several methods [12, 5, 16, 44, 27] have proposed solutions to address this challenging scenario, designing attacks to fool the popular Siamese-based trackers [24, 2, 26, 38]. Among these, however, the attacks in [12, 5, 16] are either too time-consuming or designed to work on the entire video, thus not applicable to fooling a tracker in real time and in an online fashion. By contrast, [44, 27] leverage generative methods [34, 41] to synthesize perturbations in real time and

thus can effectively attack online trackers. Nevertheless, these methods still require generating a perturbation for the search region of every individual input frame, which causes an undesirable computational burden. This burden becomes particularly pronounced for long-term trackers, because the search region grows to incorporate larger context in the presence of occlusions or when losing track of the object, leading to an increased attack time.

In this paper, we address this by introducing a temporally-transferable attack strategy. Specifically, we learn to generate a single, one-shot perturbation that is transferable across all the frames of the input video sequence. To this end, unlike existing works [44, 27] that generate perturbations for the template and for every search image independently, we train a perturbation generator to synthesize a *single* perturbation from the template, and add this perturbation to every subsequent search image. Because the perturbation is generated only once from the template, applying it to the larger search regions of long-term trackers comes at no additional cost.

Furthermore, in contrast to existing frameworks [44, 27], our approach naturally extends to performing *targeted* attacks so as to steer the tracker to follow any specified trajectory. To this end, we condition our perturbation generator on the targeted direction and train the resulting conditional generator to produce perturbations that correspond to arbitrary, diverse input directions. At test time, we can then pre-compute perturbations for a small number of diverse directions, e.g., 12 in our experiments, and apply them in turn so as to generate the desired complex trajectory. We illustrate this in Fig. 1, where a single perturbation computed in 8 ms can steer the tracker to move along a given direction for the entire video sequence, and will show more complex trajectories in our experiments.

Overall, our contributions can be summarized as follows:

- We introduce a temporally-transferable attack strategy to fool Siamese-based trackers by generating a single, one-shot perturbation from the template only.
- Our approach is applicable in an online manner, requiring only 8ms to generate our one-shot perturbation, thus maintaining real-time performance.
- Our framework naturally extends to targeted attacks, and our conditioning scheme allows us to steer the tracker to follow complex, erroneous trajectories. In practice, this would let one generate plausible incorrect tracks, making it harder to detect the attack.

We demonstrate the benefits of our approach on 4 benchmark datasets, namely, OTB100 [40], VOT2018 [20] and UAV123 [30] and VOT2018-LT [20]. Our experiments show that our approach yields more effective attacks than the state-of-the-art method [44], even though it needs to attack every individual video frame while we don't. We will make our code publicly available.

2. Related Work

Visual Object Tracking. VOT aims to estimate the position of a template cropped from the first frame of a video in each of the subsequent video frames. Unlike most other visual recognition tasks, such as image classification or object detection, that rely on predefined categories, VOT seeks to generalize to any target object at inference time. As such, early works mainly focused on measuring the correlation between the template and the search image [3], extended to exploiting multi-channel information [17] and spatial constraints [18, 8].

Nowadays, VOT is commonly addressed by end-to-end learning strategies. In particular, Siamese network-based trackers [2, 26, 38, 24, 45] have grown in popularity because of their good speed-accuracy tradeoff and generalization abilities. The progress in this field includes the design of a cross-correlation layer to compare template and search image features [2], the use of a region proposal network (RPN) [35] to reduce the number of multi-scale correlation operations [2], the introduction of an effective sampling strategy to account for the training data imbalance [45], the use of multi-level feature aggregation and of a spatially-aware sample strategy to better exploit deeper ResNet backbones [25], and the incorporation of an auxiliary segmentation training objective to improve the tracking accuracy [38]. In our experiments, we will focus on the SiamRPN++ [24] tracker due to its popularity in real-world applications. Nevertheless, we will study the transferability of our generated adversarial attacks to other trackers.

Adversarial Attacks. Adversarial attacks were first investigated in [37] to identify the vulnerability of modern deep networks to imperceptible perturbations in the context of image classification. Different attack strategies were then studied, including single step gradient descent [11, 22], DeepFool [29, 9], and computationally more expensive attacks, such as CW [4], JSMA [32], and others [7, 36, 31]. As of today, PGD [28] is considered as one of the most effective attack strategies, and has been employed to other visual recognition tasks, such as object detection [43] and semantic segmentation [1].

Inspired by this progress, iterative adversarial attacks have been studied in the context of VOT. In particular, SPARK [13] computes incremental perturbations by using information from the past frames; [6] exploits the full video sequence to attack the template by solving an optimization problem relying on a dual attention loss. While effective, most of the above-mentioned attacks are time-consuming, because of their use of heavy gradient computations or iterative schemes. As such, they are ill-suited to attack an online visual tracking system in real time. [39] also relies on a gradient-based scheme to generate a physical poster that will fool a tracker. While the attack is then real-time, it requires the attacker to physically alter the environment.

As an efficient alternative to iterative attacks, AdvGAN [42] proposed to train a generator that synthesizes perturbations in a single forward pass. Such generative perturbations were extended to VOT in [44, 27]. For these perturbations to be effective, however, both [44] and [27] proposed to attack every individual search image, by passing it through the generator. To be precise, while [44] studied the problem of attacking the template only the success of the resulting attacks was shown to be significantly lower than that of perturbing each search image. Doing so, however, becomes increasingly expensive as the search region grows, as can happen in long-term tackers. Here, we show that temporally-transferable perturbations can be generated in a one-shot manner from the template, yet still effectively fool the VOT system in every search image; the success of our approach lies in the fact that, while the perturbation is generated from the template, it is added to every search image. This addition, however, comes at virtually no cost, as it does not involve inference in a network. Furthermore, our approach can be extended to producing targeted attacks by conditioning the generator on desired directions. In contrast to [27], which only briefly studied targeted attacks in the restricted scenario of one specific pre-defined trajectory, our approach allows us to create arbitrary, complex trajectories at test time, by parametrizing them in terms of successive targeted perturbations.

3. Method

In this section, we introduce our temporally-transferable attack framework for object trackers based on Siamese networks. The goal of our method is to efficiently generate a single perturbation from the object template, which we can then transfer to adversarially attack the subsequent video frames. Below, we formalize the task of performing adversarial attacks on an object tracker, and then introduce our one-shot perturbation strategy in detail, first for untargeted attacks and then for targeted ones.

3.1. Problem Definition

Let $\mathbf{X} = \{\mathbf{X}_i\}_1^T$ denote the frames of a video sequence of length T , and \mathbf{z} be the template cropped from the first frame of the video. A tracker $\mathcal{F}(\cdot)$ aims to locate the template \mathbf{z} in search regions extracted from the subsequent video frames. Since the tracked target generally does not move wildly between two consecutive frames, most trackers only search for the target in a small region \mathbf{S}_i centered at the position obtained in the previous frame. At each time-step, the template \mathbf{z} and the search region \mathbf{S}_i are first passed individually through a shared backbone network, and the resulting features are processed by some non-shared layers and fused by depth-wise separable correlation. The fused features then act as input to a region proposal network (RPN), which predicts a classification map $\mathbf{H} \in \mathbb{R}^{H \times W \times K}$ and a bounding

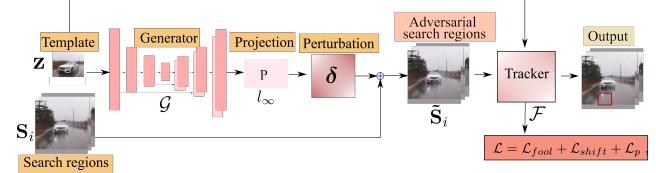


Figure 2. **Our temporally-transferable attack framework.** We generate a single perturbation from the template, and add it to the search region of every subsequent frame so as to fool the tracker.

box regression map $\mathbf{R} \in \mathbb{R}^{H \times W \times 4K}$, where K denotes the bounding box anchors at each spatial location. In short, at each spatial location, the classification map encodes the probability of each anchor to contain the target, and the regression map produces a corresponding offset for the anchor location and size. In practice, most trackers further re-rank the proposals by exploiting both a Gaussian window, which penalizes the proposals that are away from the region center, and scale change penalty, which prevents the object size from varying too strongly from that in the previous frame.

Formally, given the template \mathbf{z} , we aim to find a temporally-transferable perturbation δ that, when added to any search region \mathbf{S}_i to obtain an adversarial image $\tilde{\mathbf{S}}_i = \mathbf{S}_i + \delta$, leads to an incorrect target localization in frame i . In the untargeted attack scenario, we aim for the bounding box to simply deviate from the trajectory predicted by the unattacked tracker, thereby losing the tracked object. In the case of a targeted attack, we seek to force the center of the tracked window to follow a trajectory specified by the attacker, which can be either a simple fixed, yet arbitrary direction, or a more complicated trajectory.

3.2. One-Shot Perturbation Generator

Pipeline. Figure 2 illustrates the overall architecture of our framework, which consists of two main modules: a generator \mathcal{G} and tracker \mathcal{F} . Unlike existing works [44, 27], we train our perturbation generator to synthesize a *single* perturbation δ from the template, and simply add this perturbation to every subsequent search image. To train the generator, we therefore extract a template \mathbf{z} from the first frame of a given video sequence and feed it to the generator to obtain a perturbation δ . We then crop N search regions from the subsequent video frames using ground-truth information, and add δ to each such regions to obtain adversarial search regions $\tilde{\mathbf{S}} = \{\tilde{\mathbf{S}}_i\}_1^N$. Finally, we feed the clean template \mathbf{z} and each adversarial search region $\tilde{\mathbf{S}}_i$ to the tracker to produce an adversarial classification map $\tilde{\mathbf{H}}_i \in \mathbb{R}^{H \times W \times K}$ and regression map $\tilde{\mathbf{R}}_i \in \mathbb{R}^{H \times W \times 4K}$.

Our goal is for these adversarial classification and regression maps to fool the tracker, i.e., result in erroneously locating the target. To this end, we compute the classification map \mathbf{H}_i for the unperturbed search image, and seek to decrease the score in $\tilde{\mathbf{H}}_i$ of any proposal j such that

$\mathbf{H}_i(j) > \tau$, where $\mathbf{H}_i(j)$ indicates the probability for anchor j to correspond to the target¹ and τ is a threshold. Following [44], we achieve this by training the perturbation generator \mathcal{G} with the adversarial loss term

$$\begin{aligned} \mathcal{L}_{fool}(\mathcal{F}, \mathbf{z}, \tilde{\mathbf{S}}_i) = & \lambda_1 \sum_{j|\mathbf{H}_i(j)>\tau} \max \left(\tilde{\mathbf{H}}_i(j) - (1 - \tilde{\mathbf{H}}_i(j)), \mu_c \right) \\ & + \lambda_2 \sum_{j|\mathbf{H}_i(j)>\tau} \left(\max \left(\tilde{\mathbf{R}}_i^w(j), \mu_w \right) + \max \left(\tilde{\mathbf{R}}_i^h(j), \mu_h \right) \right), \end{aligned} \quad (1)$$

where $\tilde{\mathbf{R}}_i^w(j)$ and $\tilde{\mathbf{R}}_i^h(j)$ represent the width and height regression values for anchor j . The first term in this objective aims to simultaneously decrease the target probability and increase the background probability for anchor j where the unattacked classification map contained a high target score. The margin μ_c then improves the numerical stability of this dual goal. The second term encourages the target bounding box to shrink, down to the limits μ_w and μ_h , to facilitate deviating the tracker.

3.2.1 Untargeted Attacks

The loss \mathcal{L}_{fool} discussed above only aims to decrease the probability of the anchors obtained from the unattacked search region. Here, we propose to complement this loss with an additional objective seeking to explicitly activate a different anchor box, which we will show in our experiments to improve the attack effectiveness. Specifically, we aim for this additional loss to activate an anchor away from the search region center, so as to push the target outside the true search region, which ultimately will make the tracker be entirely lost. To achieve this, we seek to activate an anchor t lying at a distance d from the search region center. We then write the loss

$$\mathcal{L}_{shift}(\mathcal{F}, \mathbf{z}, \tilde{\mathbf{S}}_i) = \lambda_3 L_{cls}(\tilde{\mathbf{H}}_i(t)) + \lambda_4 L_{reg}(\tilde{\mathbf{R}}_i(t), \mathbf{r}^*) , \quad (2)$$

where L_{cls} is a classification loss encoding the negative log-likelihood of predicting the target at location t , and L_{reg} computes the L_1 loss between the regression values at location t and pre-defined regression values $\mathbf{r}^* \in \mathbb{R}^4$.

3.2.2 Targeted Attacks

The untargeted shift loss discussed above aims to deviate the tracker from its original trajectory. However, it does not allow the attacker to force the tracker to follow a pre-defined trajectory. To achieve this, we modify our perturbation generator to be conditioned on the desired direction we would like the tracker to predict. In practice, we input this information to the generator as an additional channel, concatenated to the template.

¹With this notation, we assume, w.l.o.g., that j represents a specific anchor at a specific spatial location.

Specifically, we compute a binary mask $\mathbf{M}_i \in \{0, 1\}^{(W \times H)}$, and set $\mathbf{M}_i(j) = 1$ at all spatial locations under the bounding box which we aim the tracker to output. Let \mathbf{B}_i^t be such a targeted bounding box, and \mathbf{r}_i^t the corresponding desired offset from the nearest anchor box. We can then express a shift loss similar to the one in Eq. 2 but for the targeted scenario as

$$\mathcal{L}_{shift}(\mathcal{F}, \mathbf{z}, \tilde{\mathbf{S}}_i, \mathbf{M}_i) = \lambda_3 L_{cls}(\tilde{\mathbf{H}}_i(t)) + \lambda_4 L_{reg}(\tilde{\mathbf{R}}_i(t), \mathbf{r}_i^t) , \quad (3)$$

where, with a slight abuse of notation, t now encodes the targeted anchor.

3.2.3 Overall Loss Function

In addition to the loss functions discussed above, we use a perceptibility loss \mathcal{L}_p aiming to make the generated perturbations invisible to the naked eye. We express this loss as

$$\mathcal{L}_p = \lambda_5 \|\mathbf{S}_i - \text{Clip}_{\{\mathbf{x}_i, \epsilon\}}\{\mathbf{S}_i + \boldsymbol{\delta}\}\|_2^2 , \quad (4)$$

where the Clip function enforces an L_∞ bound ϵ on the perturbation.

We then write the complete objective to train the generator as

$$\mathcal{L}(\mathcal{F}, \mathbf{z}, \mathbf{S}_i) = \mathcal{L}_{fool} + \mathcal{L}_{shift} + \mathcal{L}_p , \quad (5)$$

where \mathcal{L}_{shift} corresponds to Eq. 2 in the untargeted case, and to Eq. 3 in the targeted one.

3.3 Attacking the Tracker at Inference Time

Once the generator is trained using the loss in Eq. 5, we can use it to generate a temporally-transferrable perturbation from the template of any new test sequence, and use the resulting perturbation in an online-tracking phase. Generating such a perturbation takes only 8 ms, and adding it to the subsequent frames is virtually free.

For targeted attacks, in the simple case where the desired trajectory is a constant direction, we condition the generator on a specific mask \mathbf{M} , which activates a square patch centered along the desired direction and at a given offset from the center. To force the tracker to follow more complex trajectories during inference, such as following the ground-truth trajectory with an offset, we precompute perturbations for a small number, K , of predefined, diverse directions, with $K = 12$ in our experiments. We then define the desired complex trajectory as succession of these predefined directions.

4. Experiments

Datasets. Following [44], we train our perturbation generator on GOT-10K [15] and evaluate its effectiveness on 3 short-term tracking datasets, OTB100 [40], VOT2018 [20]

and UAV123 [30], and on one long-term benchmark VOT2018-LT [20]. We primarily use the SiamRPN [26] and SiamRPN++ [24] trackers with AlexNet [21] and [14] backbones, respectively, and train our generator to perform white-box attacks. We also study our attacks transfer to other popular trackers in Section 4.4.

Evaluation Metrics. As in [44, 12, 6, 27], we report the performance of our adversarial attacks using the metrics employed by each dataset to evaluate the effectiveness of unattacked trackers. Specifically, for OTB100 [40] and UAV123 [30], we report the precision (P) and success scores (S). The precision encodes the proportion of frames for which the center of the tracking window is within 20 pixels of the ground-truth center. The success corresponds to the proportion of frames for which the overlap between the predicted and ground-truth tracking window is greater than a given threshold. For VOT2018, we report the Expected Average Overlap (EAO), a measure that considers both the accuracy (A) and robustness (R) of a tracker. Specifically, the accuracy denotes the average overlap, and the robustness is computed from the number of tracking failures. Furthermore, we also report the number of restarts because the standard VOT evaluation protocol reinitializes the tracker once it is too far away from the ground truth. For VOT2018-LT [20], we report the tracker precision, recall, and F1-score. To evaluate targeted attacks, we further report the proportion of frames in which the predicted center and the target trajectory’s center are at a distance of at most 20 pixels for short-term datasets and 50 pixels for VOT2018-LT [20].

Implementation Details. We implement our approach in PyTorch [33] and perform our experiments on an NVIDIA Tesla V100 GPU with 32GB RAM. We train the generator using pre-cropped search images uniformly sampled every 10 frames from the video sequences of GOT-10K [15]. We use the Adam [19] optimizer with a learning rate of 2×10^{-4} . We set the margin thresholds m_c, m_w, m_h to -5 as in [44], and the l_∞ bound ϵ to {8, 16}. To fool the tracker, we use $\lambda_1 = 0.1, \lambda_2 = 1$ in Eq 1, and activate an anchor at distance $d = 4$ directly below the true center and of size 64×64 for all untargeted experiments. We set λ_5 to 500 for all experiments. Furthermore, we set the shift loss weights λ_3 and λ_4 to 0.1 and 1, respectively. For targeted attacks, we define r_i^t in Eq. 3 as a randomly-selected anchor at distance $d = 4$ from the true center and set its size to 64×64 for all datasets, except for VOT2018-LT where we use 90×90 . We resize the search images to 255×255 and the template to 127×127 before passing them to the tracker. For SiamRPN++[24], the final feature map will be of size $W = 25, H = 25$.

Methods. We compare our approach with the state-of-the-art, generator-based CSA [44] attack strategy, which also performs online attacks. CSA can be employed in 3

Methods.	Attack cost (ms)	SiamRPN [26]				SiamRPN++ [24]			
		$\epsilon = 8$		$\epsilon = 16$		$\epsilon = 8$		$\epsilon = 16$	
		S (\uparrow)	P (\uparrow)	S (\uparrow)	P (\uparrow)	S (\uparrow)	P (\uparrow)	S (\uparrow)	P (\uparrow)
Normal	0	65.1%	85.5%	65.1%	85.1%	69%	91%	69%	91%
CSA-T	3	54.3%	72.0%	55.0%	73.6%	60.8%	80.9%	59.0%	79.3%
CSA-S	4720	34.5%	51.4%	32.5%	45.8%	34.8%	49.3%	37.1%	53.1%
CSA-TS	4720	30.7%	46.2%	31.0%	46.8%	33.1%	48.4%	34.7%	51.0%
Ours _f	8	46.3%	66.5%	41.8%	58.8%	38.7%	54.0%	38.8%	54.9%
Ours	8	20.8%	25.7%	14.5%	17%	14.9%	18.3%	7.0%	8.0%

Table 1. Untargeted attack results on **OTB100** [40].

settings: CSA_T, which on attacks the template, CSA_S, which attacks all search images, and CSA_TS, which attacks both the template and all search regions. We denote our temporally-transferable attacks obtained from the total loss in Eq. 5 as Ours and a variant of our method without the term \mathcal{L}_{shift} but only with \mathcal{L}_{fool} as Ours_f.

Unlike CSA [44], which takes 3ms to attack the template and 8ms to compute a perturbation for each search region, our approach generates a single transferable perturbation from the template in 8ms, and uses it at virtually no additional cost² for the *entire* video sequence to attack in an untargeted manner. For targeted attacks, we precompute $K = 12$ diverse directional perturbations, each taking 8ms to generate, and use them to force the tracker to follow any target trajectory. In our experiments, we also report the average attack cost per video over the full dataset for all approaches.

4.1. Untargeted Attacks

Results on OTB100. In Table 1, we compare the results of our approach and of the different settings of CSA for both the SiamRPN and SiamRPN++ trackers. Our method works at a speed comparable to CSA_T but significantly outperforms it in all cases. Furthermore, thanks to our shift loss, our attack becomes even stronger than CSA_S and CSA_TS, while requiring only a fraction of their computation budget. For example, on SiamRPN++ [24], Ours reduces the precision of the tracker to 8%, compared to 51% for CSA_TS.

Results on VOT2018. As shown in Table 2, our approach is equally effective on VOT2018 [20]. Note that, in this dataset, the accuracy does not significantly decrease because the tracker restarts as soon as the object moves away from the ground-truth box. Importantly, unlike CSA, we compute the perturbation only from the template extracted from the first frame, even when the tracker restarts with a new template after a failure. In other words, our perturbation computed from the initial template remains effective and transferable to temporally-distance search regions. As shown in Table 2, for SiamRPN++ [24] with $\epsilon = 16$, our approach significantly decreases the EAO, which is the primary metric to rank trackers, from 0.261 to 0.020 while increasing the number of restarts from 94 to 1249. In Fig-

²The additional cost comes from adding the perturbation to the search images, an operation that CSA_S and CSA_TS must also perform.

Method	$\epsilon = 8$						$\epsilon = 16$					
	Attack cost (ms)	A(↑)	R(↓)	EAO(↑)	Re(↓)	Attack cost (ms)	A(↑)	R(↓)	EAO(↑)	Re(↓)		
SiamRPN++ [24]	Normal	0	0.60	0.32	0.340	69	0	0.60	0.32	0.34	69	
	CSA-T	9	0.56	0.57	0.215	121	15	0.58	1.06	0.133	226	
	CSA-S	2832	0.49	1.99	0.074	423	2832	0.48	2.20	0.067	464	
	CSA-TS	2832	0.46	1.70	0.086	363	2856	0.49	1.98	0.075	421	
SiamRPN [26]	Ours _f	8	0.55	1.55	0.095	330	8	0.50	1.80	0.079	384	
	Ours	8	0.55	7.14	0.017	1524	8	0.52	7.82	0.014	1669	
	Normal	0	0.57	0.44	0.261	94	0	0.57	0.44	0.261	94	
	CSA-T	15	0.56	1.04	0.132	222	15	0.56	1.10	0.126	234	
SiamRPN [26]	CSA-S	2832	0.45	1.76	0.080	376	2832	0.46	1.95	0.077	417	
	CSA-TS	2853	0.43	1.63	0.082	348	2856	0.43	1.90	0.076	405	
	Ours _f	8	0.51	1.45	0.103	310	8	0.50	1.36	0.110	291	
	Ours	8	0.55	5.06	0.027	1080	8	0.53	5.85	0.020	1249	

Table 2. Untargeted attack results on VOT2018 [20].

ure 3, we visually compare the results of our approach and of the baselines in different conditions, defined as attribute values in the dataset. Ours generalizes nicely to all these scenarios.

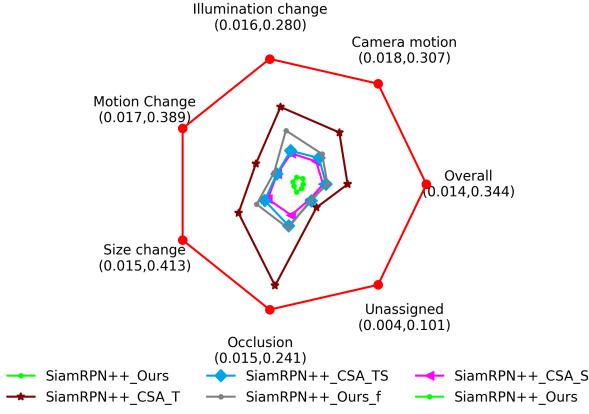


Figure 3. Quantitative analysis of the influence of different attributes on VOT2018 [20] with and without attacks. The closer a point is to the center, the least successful the tracker is.

Results on UAV123. In Figure 4, we provide success and precision plots obtained by varying the overlap and location error thresholds, respectively, with $\epsilon = 16$. We observe a similar trend here: Ours_f yields results that are comparable to those of CSA_TS despite being much faster, and Ours significantly outperforms all the baselines at all the thresholds. For example, at a 20 pixel threshold, while CSA_TS yields a tracking precision of 0.504 precision, Ours decreases it to 0.139.

Results on VOT2018-LT. VOT2018-LT [20] is dedicated to long-term tracking, and thus evaluating the strength of an attack should take into account not only the performance metrics but also the speed of the tracker. This is because, when the detection confidence is less than 0.8 the tracker falls back to using a larger search region of 850×850 and thus becomes slower. This also means that the attack should not only minimize the probability of the true target but also activate adversarial boxes with the confidence larger than 0.8 so that search images are not resized.

As can be seen from Table 3, Ours yields F1 scores that

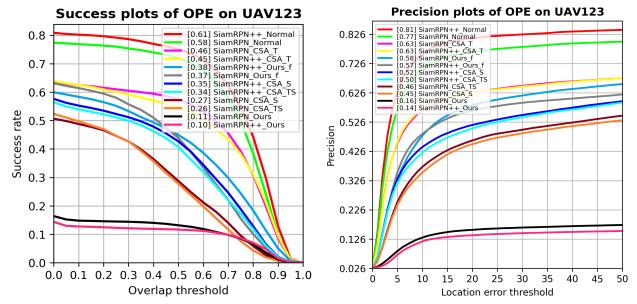


Figure 4. Untargeted attack results on UAV123 [30].

Networks	Attack cost (ms)	DASiamRPN_LT [26]				SiamRPN++_LT [24]			
		$\epsilon = 8$		$\epsilon = 16$		$\epsilon = 8$		$\epsilon = 16$	
		F1(↑)	fps(↑)	F1(↑)	fps(↑)	F1(↑)	fps(↑)	F1(↑)	fps(↑)
Normal	0	0.48	23.3	0.48	23.3	0.61	16.3	0.61	16.3
CSA_T	3	0.41	20.3	0.42	27.2	0.47	13.6	0.28	8.5
CSA_S	33552	0.38	27.2	0.37	20.3	0.48	6.4	0.46	7.1
CSA_TS	33552	0.39	16.3	0.38	16.5	0.47	8.1	0.46	7.4
Ours _f	8	0.41	20.3	0.42	20.4	0.53	9.0	0.50	6.6
Ours	8	0.35	20.3	0.26	20.5	0.18	12.4	0.08	16.3

Table 3. Untargeted attack results on VOT2018-LT [20].

are similar to those of CSA_T, but lower frame-rates, showing that it tends to produce confidences that are lower than 0.8. By contrast, Ours not only decreases the F1 score to significantly lower values than all the baselines, but it also maintains high frame-rates.

4.2. Targeted attacks

Let us now turn to targeted attacks. In this context, we consider two scenarios:

1. The attacker forces the tracker to follow a fixed direction. We illustrate this with 4 different directions, consisting of shifting the box by $(\pm 3, \pm 3)$ pixels in each consecutive frames, corresponding to the four directions $+45^\circ, -45^\circ, +135^\circ, -135^\circ$.
2. The attacker seeks for the tracker to follow a more complicated trajectory. To illustrate this, we force the tracker to follow the ground-truth trajectory with a fixed offset (Δ_x, Δ_y) . We set this offset to $(\pm 80, \pm 80)$ for the short-term tracking datasets and to $(\pm 150, \pm 150)$ for VOT2018-LT to account for the larger resolution of its videos.

In both cases, we pre-compute perturbations corresponding $K = 12$ diverse directions and use them to force the tracker to follow the target trajectory. To this end, we first sample K points at a distance $d = 4$ from the true object center and, for each, synthesize a condition mask $M_i \in \{0, 1\}^{(W \times H)}$ whose active region is centered at the sampled point. We then feed each such mask with the template to obtain directional perturbations, which we will then transfer to the search images. At each frame, we compute the direction



Figure 5. **Perturbed search regions.** We visualize adversarial search regions for targeted attacks obtained with our framework for $\epsilon = 16$ with SiamRPN++ [24].

	Datasets	Normal	$+45^\circ$	-45°	$+135^\circ$	-135°	Avg.
Direction-based targets	OTB100 [40]	91%	57.7%	47.5%	55.0%	54.7%	53.7%
	VOT2018 [20]	72.8%	68.2%	71.0%	67.5%	69.9%	69.2%
	UAV123 [30]	81.3%	43.1%	41.0%	49.2%	47.3%	45.2%
	VOT2018-LT [20]	77.2%	35.6%	35.1%	35.7%	35.0%	35.3%
	Datasets	Normal	(Δ_x, Δ_y)	$(\Delta_x, -\Delta_y)$	$(-\Delta_x, +\Delta_y)$	$(-\Delta_x, -\Delta_y)$	Avg.
Offset-based targets	OTB100 [40]	91%	50.7%	45.4%	44.2%	42.0%	45.6%
	VOT2018 [20]	72.8%	55.2%	53.9%	58.2%	57.5%	56.2%
	UAV123 [30]	81.3%	36.2%	38.4%	44.1%	40.9%	39.9%
	VOT2018-LT [20]	77.2%	25.7%	25.1%	28.5%	21.1%	25.1%

Table 4. **Targeted attack results.** We report precision scores with $\epsilon = 16$. $\Delta_x \in \{80, 150\}$ and $\Delta_y \in \{80, 150\}$ depending on the dataset.

the tracker should move in and use the precomputed perturbation that is closest to this direction.

We report the results for our two attack scenarios in Table 4 for all datasets with the SiamRPN++ [24] tracker. Since the goal is to follow the desired trajectory, we report the precision for a location threshold of 20 pixels for all short-term datasets and 50 pixels for the long-term one. For direction-based targets, our perturbations allow us to effectively follow the target trajectory. For example, on OTB100 [40], our approach yields a precision of 53.7% on average on the 4 directions, while the tracker has an original precision of 91% on the original data. Similarly, on VOT2018, our targeted attacks achieve a precision of 69.2% on average, compared to the original tracker’s precision of 72.8%. For offset-based targets, which are more challenging than direction-based ones, our approach yields precision scores of 45.6%, 56.2%, 30.9% and 24.9% on OTB100, VOT2018, UAV, and VOT2018-LT, respectively.

Figure 6 shows the results of targeted attacks on various datasets with SiamRPN++ [24]. The results on the left, where the tracker follows the ground-truth trajectory with an offset, illustrate the real-world applicability of our attacks, where one could force the tracker to follow a realistic, yet erroneous path. Such realistic trajectories can deceive the system without raising any suspicion.

In Figure 5, we visualize adversarial search regions obtained with different l_∞ bound values for targeted attacks. We observe that, for $\epsilon = 16$, our results remain imperceptible thanks to our similarity loss. We provide more qualitative results and perceptibility analyses in the supplementary material.

\mathcal{L}_{fool}^{cls}	\mathcal{L}_{fool}^{reg}	$\mathcal{L}_{shift}^{cls}$	$\mathcal{L}_{shift}^{reg}$	OTB100 [40]		VOT2018 [20]	
				Success(\uparrow)	Precision(\uparrow)	Success(\uparrow)	EAO(\uparrow)
-	-	-	-	69.5%	90.5%	0.340	
✓	-	-	-	58.8%	66.5%	0.088	
-	✓	-	-	50.6%	72.6%	0.128	
-	-	✓	-	50.1%	65.8%	0.094	
-	-	-	✓	65.4%	86.3%	0.024	
✓	✓	-	-	38.7%	54.0%	0.095	
-	-	✓	✓	40.5%	52.7%	0.063	
✓	-	✓	-	19.1%	23.8%	0.024	
-	✓	-	✓	46.4%	63.1%	0.092	
✓	✓	✓	✓	14.9%	18.3%	0.017	

Table 5. **Ablation study.** Contribution of each loss for untargeted attacks with SiamRPN++ [24]. We set $\epsilon = 8$.

Shift d	OTB100 [40]		VOT2018 [20]			
	Success(\uparrow)	Precision(\uparrow)	A(\uparrow)	R(\downarrow)	EAO(\uparrow)	Re(\downarrow)
0	38.7%	54.0%	0.549	1.54	0.095	330
2	16.6%	20.5%	0.558	5.25	0.026	1180
4	14.9%	18.3%	0.552	7.31	0.017	1524
6	13.8%	17.5%	0.564	7.27	0.017	1552
8	15.7%	19.6%	0.615	7.32	0.018	1563
10	22.6%	28.7%	0.604	5.57	0.027	1189

Table 6. **Ablation study.** Effect of d in \mathcal{L}_{shift} for untargeted attacks with SiamRPN++ [24]. We set $\epsilon = 8$.

4.3. Ablation Study

In this section, we analyze the impact of each loss term of our framework. To do so, we use the SiamRPN++ tracker [24] and perform untargeted attacks. In Table 5, we report our results on OTB100 [40] and VOT2018 [20] with different combination of loss terms, where \mathcal{L}_{fool}^{cls} and \mathcal{L}_{fool}^{reg} represent the classification and regression components of the fooling loss of Eq. 1, $\mathcal{L}_{shift}^{cls}$ and $\mathcal{L}_{shift}^{reg}$ represent the same terms for the shift loss of Eq. 2. To summarize, while all loss terms are beneficial, the classification-based terms are more effective than regression-based ones. For example, using either \mathcal{L}_{fool}^{cls} or \mathcal{L}_{fool}^{cls} has more impact than \mathcal{L}_{fool}^{reg} or \mathcal{L}_{fool}^{reg} . In Table 6, we study the impact of the shift distance d in Eq. 2 on the performance of untargeted attacks. For a feature map of size 25×25 for SiamRPN++ [24], the performance of our approach is stable for a drift in the range 4 to 8. However, for $d = 2$, our attacks have less effect on the tracker, and for $d = 10$, the influence of the attack decreases because of the Gaussian prior used by the tracker.

4.4. Transferability Analysis

To evaluate the transferability of our attacks, we apply the perturbation-generator trained on SiamRPN++ [24] with a ResNet [14] backbone to three other state-of-the-art trackers: AlexNet-based SiamFC [2], ResNet-based SiamMask [38] and a similar SiamRPN++(M) [24] but with a MobileNet backbone. As can be seen in Table 7, our approach transfers better than CSA_T to SiamMask and SiamRPN++(M). However, neither CSA nor our approach generalize well to SiamFC, which we conjecture to be due to SiamFC using a different backbone from that of SiamRPN++ [24] and to its non-RPN-based architecture.



Figure 6. **Qualitative results.** (a) Results of targeted attacks where the tracker is forced to follow the ground-truth with a fixed offset. (b) The tracker is forced to move in a constant direction, indicated by the red arrow.

We therefore believe that generating black-box attacks that generalize to arbitrary architectures would be an interesting avenue for future research.

Tracker	Method	VOT2018 [20]			OTB100++ [40]	
		A(\uparrow)	R(\downarrow)	EAO(\uparrow)	Re(\downarrow)	S (\uparrow)
SiamMask [38]	Normal	0.59	0.25	0.407	53	64.7%
	CSA_T	0.53	0.36	0.290	77	57.3%
	CSA_S	0.35	2.81	0.045	601	21.5%
	CSA_TS	0.34	2.87	0.046	612	20.2%
	Ours	0.46	2.46	0.056	516	29.4%
	Ours _{shift}	0.57	6.08	0.023	1298	16.0%
SiamRPN++(M) [24]	Normal	0.59	0.24	0.398	51	65.1%
	CSA_T	0.56	0.45	0.265	95	61.5%
	CSA_S	0.41	2.20	0.066	471	28.1%
	CSA_TS	0.40	2.17	0.067	465	26.3%
	Ours	0.48	1.72	0.085	367	34.7%
	Ours _{shift}	0.52	5.15	0.028	1100	21.7%
SiamFC [2]	Normal	0.49	0.55	0.211	118	56.9%
	CSA_T	0.48	0.52	0.213	112	56.8%
	CSA_S	0.49	0.56	0.210	120	57.2%
	CSA_TS	0.49	0.57	0.211	121	57.5%
	Ours	0.48	0.55	0.201	118	57.7%
	Ours _{shift}	0.49	0.59	0.200	126	55.9%

Table 7. **Transferability analysis** on OTB100 [40] with a generator trained on SiamRPN++ [24]. We set $\epsilon = 8$.

5. Conclusion

We have presented a one-shot temporally-transferable perturbation framework to efficiently attack a VOT algorithm using only template information. Our generates a single perturbation from the object template in only 8ms, which can then be transferred to attack the subsequent frames of the video. Furthermore, we have demonstrated that conditioning the generator allows us to steer the tracker to follow any specified trajectory by precomputing a few diverse directional perturbations. We believe that our work highlights the vulnerability of object trackers and will motivate the researchers to design robust defense mechanisms in the future.

References

- [1] Anurag Arnab, Ondrej Miksik, and Philip HS Torr. On the robustness of semantic segmentation models to adversarial attacks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 888–897, 2018. **3248**, **3249**

- [2] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016. [3248](#), [3249](#), [3254](#), [3255](#)
- [3] David S Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 2544–2550. IEEE, 2010. [3249](#)
- [4] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017. [3249](#)
- [5] Xuesong Chen, Xiyu Yan, Feng Zheng, Yong Jiang, Shu-Tao Xia, Yong Zhao, and Rongrong Ji. One-shot adversarial attacks on visual tracking with dual attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10176–10185, 2020. [3248](#)
- [6] Xuesong Chen, Xiyu Yan, Feng Zheng, Yong Jiang, Shu-Tao Xia, Yong Zhao, and Rongrong Ji. One-shot adversarial attacks on visual tracking with dual attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10176–10185, 2020. [3249](#), [3252](#)
- [7] Francesco Croce and Matthias Hein. Sparse and imperceptible adversarial attacks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4724–4732, 2019. [3249](#)
- [8] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. Learning spatially regularized correlation filters for visual tracking. In *Proceedings of the IEEE international conference on computer vision*, pages 4310–4318, 2015. [3249](#)
- [9] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018. [3249](#)
- [10] Volker Fischer, Mummadri Chaithanya Kumar, Jan Hendrik Metzen, and Thomas Brox. Adversarial examples for semantic image segmentation. *arXiv preprint arXiv:1703.01101*, 2017. [3248](#)
- [11] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. [3248](#), [3249](#)
- [12] Qing Guo, Xiaofei Xie, Felix Juefei-Xu, Lei Ma, Zhongguo Li, Wanli Xue, Wei Feng, and Yang Liu. Spark: Spatial-aware online incremental attack against visual tracking. *arXiv preprint arXiv:1910.08681*, 2019. [3248](#), [3252](#)
- [13] Qing Guo, Xiaofei Xie, Felix Juefei-Xu, Lei Ma, Zhongguo Li, Wanli Xue, Wei Feng, and Yang Liu. Spark: Spatial-aware online incremental attack against visual tracking. *arXiv preprint arXiv:1910.08681*, 2019. [3249](#)
- [14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. [3252](#), [3254](#)
- [15] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. [3251](#), [3252](#)
- [16] Yunhan Jia, Yantao Lu, Junjie Shen, Qi Alfred Chen, Zhenyu Zhong, and Tao Wei. Fooling detection alone is not enough: First adversarial attack against multiple object tracking. *arXiv preprint arXiv:1905.11026*, 2019. [3248](#)
- [17] Hamed Kiani Galoogahi, Terence Sim, and Simon Lucey. Multi-channel correlation filters. In *Proceedings of the IEEE international conference on computer vision*, pages 3072–3079, 2013. [3249](#)
- [18] Hamed Kiani Galoogahi, Terence Sim, and Simon Lucey. Correlation filters with limited boundaries. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4630–4638, 2015. [3249](#)
- [19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. [3252](#)
- [20] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Pflugfelder, Luka Cehovin Zajc, Tomas Vojir, Goutam Bhat, Alan Lukezic, Abdelrahman Eldekokey, et al. The sixth visual object tracking vot2018 challenge results. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018. [3249](#), [3251](#), [3252](#), [3253](#), [3254](#), [3255](#)
- [21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. [3252](#)
- [22] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016. [3248](#), [3249](#)
- [23] Kuan-Hui Lee and Jenq-Neng Hwang. On-road pedestrian tracking across multiple driving recorders. *IEEE Transactions on Multimedia*, 17(9):1429–1438, 2015. [3248](#)
- [24] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4282–4291, 2019. [3248](#), [3249](#), [3252](#), [3253](#), [3254](#), [3255](#)
- [25] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4282–4291, 2019. [3249](#)
- [26] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8971–8980, 2018. [3248](#), [3249](#), [3252](#), [3253](#)
- [27] Siyuan Liang, Xingxing Wei, Siyuan Yao, and Xiaochun Cao. Efficient adversarial attacks for visual object tracking. *arXiv preprint arXiv:2008.00217*, 2020. [3248](#), [3249](#), [3250](#), [3252](#)
- [28] 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. [3249](#)

- [29] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016. [3249](#)
- [30] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for uav tracking. In *European conference on computer vision*, pages 445–461. Springer, 2016. [3249, 3252, 3253, 3254](#)
- [31] Nina Narodytska and Shiva Kasiviswanathan. Simple black-box adversarial attacks on deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1310–1318. IEEE, 2017. [3249](#)
- [32] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016. [3249](#)
- [33] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. [3252](#)
- [34] Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge Beßongie. Generative adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4422–4431, 2018. [3248](#)
- [35] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016. [3249](#)
- [36] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019. [3249](#)
- [37] 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. [3249](#)
- [38] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip HS Torr. Fast online object tracking and segmentation: A unifying approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1328–1338, 2019. [3248, 3249, 3254, 3255](#)
- [39] Rey Reza Wiyatno and Anqi Xu. Physical adversarial textures that fool visual object tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4822–4831, 2019. [3249](#)
- [40] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2411–2418, 2013. [3249, 3251, 3252, 3254, 3255](#)
- [41] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*, 2018. [3248](#)
- [42] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*, 2018. [3250](#)
- [43] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1369–1378, 2017. [3248, 3249](#)
- [44] Bin Yan, Dong Wang, Huchuan Lu, and Xiaoyun Yang. Cooling-shrinking attack: Blinding the tracker with imperceptible noises. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 990–999, 2020. [3248, 3249, 3250, 3251, 3252](#)
- [45] Zheng Zhu, Qiang Wang, Bo Li, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 101–117, 2018. [3249](#)