# Video-Agnostic Perturbations: Efficient Targeted Attacks for Siamese Visual Tracking

**PaperID: 2266**

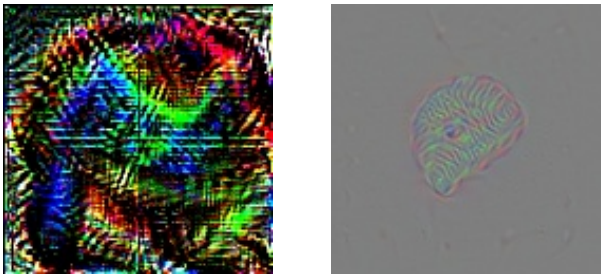


Figure 1: Left: search patch. Right: template disturb.


## Abstract

Siamese trackers are shown to be vulnerable to adversarial attacks recently. However, the existing attack methods craft the perturbations for each video independently, which comes at a non-negligible computational cost. The question is what if we can not get access to the limited computational resources in the real-world online-tracking phase.

In this paper, we show the existence of video-agnostic perturbations that can enable the targeted attack, e.g., forcing a tracker to follow the ground-truth trajectory with specified offsets, to be universal and free from inference in a network. Specifically, we attack a tracker by adding a universal imperceptible perturbation to the template image and pasting a *fake target*, i.e., a small universal adversarial patch, into the search images adhering to the predefined trajectory, so that the tracker outputs the location and size of the *fake target* instead of the real target. Our approach allows perturbing a novel video to come at no additional cost except the mere addition and pasting operations – and not require gradient optimisation or network inference. Experimental results on several datasets demonstrate that our approach can effectively fool the Siamese trackers in a targeted attack manner. We will make our code publicly available.


## 1 Introduction

Object tracking refers to the task of sequentially locating a specified moving object in a video, given only its initial state. Recently, Siamese networks [Danelljan *et al.*, 2019] [Bertinetto *et al.*, 2016] have demonstrated a significant improvement in object tracking performances. Siamese trackers formulate the visual object tracking problem as learning cross-correlation similarities between a target template and a search region. Tracking is then performed by finding the target object from the search image region by computing the highest visual similarity.

Recently, the robustness of Siamese trackers have recently attracted a lot of attention [Nakka and Salzmann, 2020] [Liang *et al.*, 2020] [Guo *et al.*, 2020]. Despite the impressive performance of Siamese architectures on challenging visual object tracking task, these trackers were shown to be highly vulnerable to perturbations. [Nakka and Salzmann, 2020] 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 to adversarially attack the subsequent video frames. [Liang *et al.*, 2020] present an end-to-end network FAN (Fast Attack Network) that uses a novel drift loss combined with the embedded feature loss to attack the Siamese network based trackers. [Guo *et al.*, 2020] propose the spatial-aware online incremental attack (a.k.a. SPARK) that performs spatial-temporal sparse incremental perturbations online and makes the adversarial attack less perceptible.

All the above methods can fool object trackers successfully. However, these attack methods are not suitable for the real-world online-tracking phase where we can not get access to the limited computational resources. The reason is, visual object tracking is a computationally intensive task which usually needs to be run in real time, but the computational source of small platforms (e.g., mobile, robotic, self-driving car) used to deploy the trackers is often limited. Therefore, the attack approaches for object tracking should occupy as little computing resources as possible. But the reality is that most attack algorithms either perform iterative optimisation to calculate adversarial examples, or generate perturbations through Generative Adversarial Networks which occupy the GPU memory.

All the above methods can fool object trackers successfully. However, these attack methods craft the perturba-

tions for each video independently, which comes at a non-negligible cost. Can we find the universal perturbations that fool a state-of-the-art deep Siamese trackers on all natural videos? We show in this paper the existence of such universal perturbations that cause the tracker to track object along any specified trajectory. Specifically, we aim to attack the tracker by adding an imperceptible perturbation to the template image and pasting a *fake target*, i.e., a small adversarial patch into the search image adhering to the predefined trajectory, so that tracker outputs the location/size of the fake target instead of the real target. Different from the existing attack methods which are intrinsic dependent on videos, we generate the perturbations that can fool the network on most natural videos. Perturbing a new target then only involves the mere addition/paste operation of the universal perturbations to the template/search image – and does not require solving an optimisation problem/gradient computation. Overall, our contributions can be summarized as follows:

- We show the existence of universal video-agnostic perturbations for state-of-the-art deep Siamese trackers. To the best of our knowledge, this is the first work to perform universal targeted attack against the visual object tracking task.
- We propose an end-to-end pipeline to generate the universal perturbations using large scale object tracking dataset.
- Unlike most existing attack methods which only attack the classification branch of the tracker, we simultaneously manipulate the classification and bounding box regression result of the Siamese tracker.
- Experiment results on OTB2015 [Wu *et al.*, 2013], GOT-10k [Huang *et al.*, 2019] and LaSOT [Huang *et al.*, 2019] datasets demonstrate that our method can effectively fool the Siamese trackers in a targeted attack manner, which comes at virtually no cost.

# 2 Related Work

## 2.1 Visual Object Tracking

Visual object tracking is a fundamental problem in computer vision, estimating the position of a template cropped from the first frame of a video in each of the subsequent video frames. The state-of-the-art trackers can be roughly summarized to three categories, including correlation filter-based, classification & updating-based and Siamese network-based trackers. Recently, Siamese networks [Li *et al.*, 2018] [Li *et al.*, 2019] have demonstrated a significant improvement in object tracking performances. Siamese trackers formulate the visual object tracking problem as learning cross-correlation similarities between a target template and a search region. Tracking is then performed by finding the target object from the search image region by computing the highest visual similarity. In particular, SiamRPN [Li *et al.*, 2018] consists of Siamese subnetwork for feature extraction and region proposal subnetwork including the classification branch and regression branch. In the inference phase, the proposed framework is formulated as a local one-shot detection task. SiamRPN++ [Li *et al.*, 2019] breaks the restriction of strict translation invariance through a simple yet effective spatial aware sampling strategy and successfully train a ResNet-driven Siamese tracker with significant performance gain. SiamFC++ [Xu *et al.*, 2020] is designed by introducing both classification and target state estimation branch, classification score without ambiguity, tracking without prior knowledge, and estimation quality score. In our experiments, we will focus on the SiamFC++ tracker due to its popularity in real-world applications. Nevertheless, we will study the transferability of our generated adversarial attacks to other trackers.

## 2.2 Adversarial Attack

Adversarial attacks were first investigated in [Szegedy *et al.*, 2013] to identify the vulnerability of modern deep networks to imperceptible perturbations in the context of image classification. Recent studies also emerge to investigate the adversarial attacks to other diverse types of tasks, e.g., natural language processing [Ren *et al.*, 2019], speech recognition [Qin *et al.*, 2019] and object detection [Wei *et al.*, 2019]. Scenarios of possible adversarial attacks can be categorized along different dimensions.

**Imperceptible Perturbation v.s. Adversarial Patch**
The imperceptible perturbations most commonly modify each pixel by on a small amount and can be found using a number of optimisation strategies such as L-BFGS [Szegedy *et al.*, 2013], Fast Gradient Sign Method (FGSM) [Goodfellow *et al.*, 2014], and PGD [Madry *et al.*, 2017]. Different from the imperceptible perturbation, the adversarial patch is extremely salient to a neural network. The adversarial patch can be placed anywhere into the input image, and causes the network to misbehave. To the best of our knowledge, we are the first to attack object trackers utilizing both the imperceptible perturbation and the adversarial patch together, which are jointly trained in an end-to-end manner. Note that our adversarial patch works in the network domain instead of the image domain. In the network-domain case, the noise is allowed to take any value and is not restricted to the dynamic range of images, while in the image domain case the noise is kept to the dynamic-range of images.

**Non-Targeted Attack v.s. Targeted Attack**
In the case of non-targeted attack, the adversary's goal is to cause the network to predict any incorrect label and the specific incorrect label does not matter. In the case of targeted attack, the adversary aims to change the network's prediction to some specific target label. For the video object tracking task, the targeted attack aims to intentionally drive trackers to output desired object positions specified by the targeted trajectory.

**Non-Universal Attack v.s. Universal Attack**
In the case of non-universal attack, the perturbations are specifically crafted of each data point independently, while the universal attack seeks a single perturbation that fools the network on most data points. In this paper, we perform the universal attack for Siamese tracker.

## 2.3 Adversarial Attack in Object Tracking

Recently, there are several explorations of the adversarial attacks on the visual object tracking task. For example, PAT

[Wiyatno and Xu, 2019] generates physical adversarial textures visa white-box attack to let the tracker lock on the texture when a tracked object moves in front of it. However, PAT validates its method by attacking a light deep regression tracker, i.e., GOTURN [Held *et al.*, 2016] that has low tracking accuracy on modern benchmarks. In this paper, we aim to attack the state-of-the-art trackers based on the Siamese networks. RTAA [Jia *et al.*, 2020] takes temporal motion into consideration when generating lightweight perturbations over the estimated tracking results frame-by-frame. However, RTAA only performs the untargeted attack for tarckers, which causes the adversarial trajectory to deviate from the original trajectory of an object. In this paper, we focus on the more challenging targeted attack for object tracking. SPARK [Guo *et al.*, 2020] computes incremental perturbations by using information from the past frames to perform targeted attack against Siamese trackers. However, SPARK needs to generate the adversarial example for every search image in a video, which is time-consuming and ill-suited to attack an online visual tracking system in real time. In this paper, we propose universal perturbations which allow perturbing a novel video to come at no additional cost except the mere addition and pasting operations.

# 3 Method

In this section, we introduce our universal targeted attack framework for object trackers based on Siamese networks. We aim to attack the tracker by adding an imperceptible perturbation to the template image and pasting a *fake target*, i.e., an adversarial patch into the search image adhering to the predefined trajectory, so that the tracker outputs the location of the fake target instead of the real target. Below, we formalize the task of performing universal targeted attacks on an object tracker, and then introduce our perturbation strategy in detail.

## 3.1 Problem Definition

Let $V = \{I_i\}_1^T$ denote the frames of a video sequence of length $T$. $B^{gt} = \{b_i^{gt}\}_1^T$ is used to represent the target's ground-truth position in each frame. The visual object tracking will predict the position $B^{pred} = \{b_i^{pred}\}_1^T$ of this target in the subsequent frames when given its initial state. In SiamFC++ [Xu *et al.*, 2020], the tracker first transforms the reference frame $I_1$ and annotation $b_1^{gt}$ to get an template image $\mathbf{z}$, and transforms the candidate frame $I_i$ to get the search image $\mathbf{x}_i$ centered at the position obtained in the previous frame. At each time-step, the template image $\mathbf{z}$ and the search image $\mathbf{x}_i$ 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 head network, which predicts a classification map $\mathbf{C}$, a bounding box regression map $\mathbf{R}$, and a quality assessment map $\mathbf{Q}$. In short, $\mathbf{C}$ encodes the probability of each spatial position to contain a the target, $\mathbf{R}$ regresses the bounding box position of the target, and $\mathbf{Q}$ predicts the target state estimation quality. The final bounding box is then generated according to $\mathbf{C}$, $\mathbf{R}$ and $\mathbf{Q}$.

Formally, we aim to train an imperceptible perturbation $\delta$ for the template image $\mathbf{z}$, and an adversarial patch $p$ for the search image $\mathbf{x}$. After adding $\delta$ to $\mathbf{z}$ and pasting the fake target $p$ to $\mathbf{x}$, the tracker outputs the location of the adversarial patch instead of the real target. Both $\delta$ and $p$ are universal (i.e., video-agnostic), which means perturbing a new video only involves the mere addition/paste of the perturbations to the template/search image – and does not require solving an optimisation problem/gradient computation.

## 3.2 Generating Universal Perturbations

In this subsection, we show how to train the universal perturbations $(\delta, p)$ for Siamese trackers. During the $k$-th iteration of training, a video $V = \{I_i\}_1^T$ is randomly select from the training dataset $\mathcal{V}$. Assuming the template perturbation at the $k$-th iteration is $\delta_k \in \mathbb{R}^{127\times127\times3}$, and the adversarial patch is $p_k \in \mathbb{R}^{128\times128\times3}$. We first randomly pick two frames $I_t, I_s$ from $V$. The clean template image $\mathbf{z} \in \mathbb{R}^{127\times127\times3}$ is generated according to $I_t$ and $b_t^{gt}$, and the perturbed template image is:

$$\tilde{\mathbf{z}} = \mathbf{z} + \delta_i. \tag{1}$$

Similarly, the clean search image $\mathbf{x} \in \mathbb{R}^{303\times303\times3}$ is generated according to $I_s$ and $b_s^{gt}$. As mentioned before, the patch is regarded as a fake target and pasted on the search image. We force the center position of the fake target to near the center position of the real target (i.e., the center of the search image) with an offset $(\Delta_x, \Delta_y)$. We set this offset to $(\pm64, \pm64)$. The width/height of the fake target is randomly selected between 32 pixels and 128 pixels. The perturbed search image is generated as follows:

$$\tilde{\mathbf{x}} = A(\mathbf{x}, p_i, (l^x, l^y), (w, h)), \tag{2}$$

where $(l_x, l_y)$ and $(w, h)$ represents the position and size of the fake target relative to the search image, respectively. $A$ is patch application operator [Brown *et al.*, 2017] which first resizes the patch $p_k \in \mathbb{R}^{128\times128\times3}$ to $\hat{p}_k \in \mathbb{R}^{w\times h\times3}$, and then applies the resized patch $\hat{p}_k$ to the search image $\mathbf{x}$ at location $(l_x, l_y)$.

Subsequently, the SiamFC++ tracker $\phi(\cdot)$ takes $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{z}}$ as input and makes predictions as follows:

$$\mathbf{C}, \mathbf{R}, \mathbf{Q} = \phi(\tilde{\mathbf{x}}, \tilde{\mathbf{z}}), \tag{3}$$

**Training Objective**

The loss function is calculated as follows:

$$\begin{aligned} L = & \frac{\alpha}{N_{\text{pos}}} \sum_{x,y} L_{\text{cls}} \left(\mathbf{C}_{x,y}, \mathbf{C}^*_{x,y}\right) \\ & + \frac{\beta}{N_{\text{pos}}} \sum_{x,y} \mathbf{1}_{\{\mathbf{C}^*_{x,y}>0\}} L_{\text{quality}} \left(\mathbf{Q}_{x,y}, \mathbf{Q}^*_{x,y}\right) \\ & + \frac{\gamma}{N_{\text{pos}}} \sum_{x,y} \mathbf{1}_{\{\mathbf{C}^*_{x,y}>0\}} L_{\text{reg}} \left(\mathbf{R}_{x,y}, \mathbf{R}^*_{x,y}\right) \\ & + \eta \cdot ||\mathbf{z}||_2^2 + \sigma \cdot ||\mathbf{x}||_2^2 \end{aligned} \tag{4}$$

where $\mathbf{C}_{x,y}, \mathbf{R}_{x,y}, \mathbf{Q}_{x,y}$ represent the values of $\mathbf{C}, \mathbf{R}, \mathbf{Q}$ at location $(x, y)$, respectively. $\mathbf{C}^*, \mathbf{R}^*, \mathbf{Q}^*$ are the fake labels generated according to the position/size of the fake target.$\mathbf{1}$ is the indicator function that takes 1 if the condition in subscribe

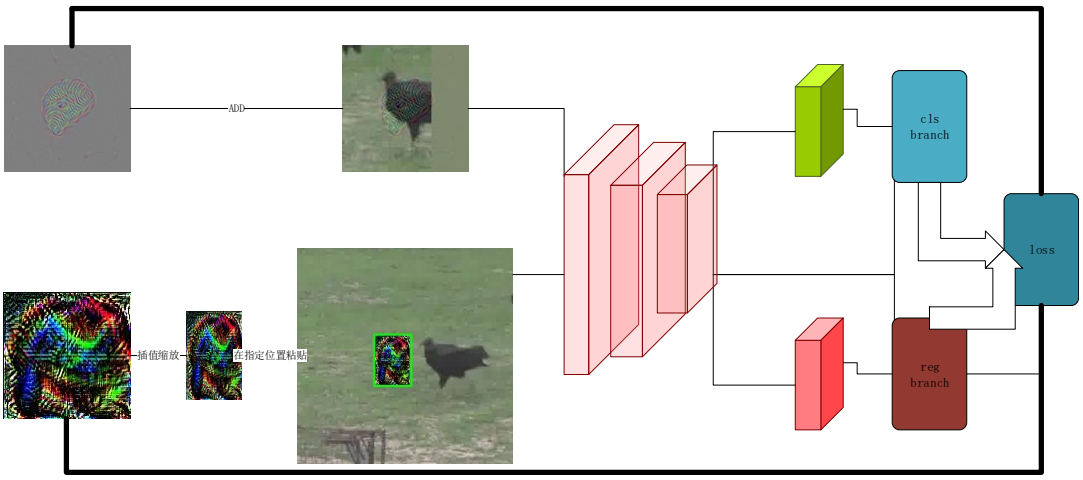



Figure 2: network structure

| iter num | 0 | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 | 8192 | 16384 | 32768 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FGT-AO | 0.004 | 0.002 | 0.002 | 0.002 | 0.002 | 0.002 | 0.003 | 0.007 | 0.042 | 0.299 | 0.668 | 0.746 | 0.781 | 0.798 | 0.820 | 0.821 | 0.818 |
| FGT-SR-50 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.001 | 0.005 | 0.044 | 0.335 | 0.749 | 0.822 | 0.855 | 0.872 | 0.895 | 0.897 | 0.890 |
| GT-AO | 0.760 | 0.757 | 0.756 | 0.757 | 0.757 | 0.758 | 0.759 | 0.753 | 0.720 | 0.474 | 0.150 | 0.095 | 0.071 | 0.041 | 0.032 | 0.032 | 0.035 |
| GT-SR-50 | 0.897 | 0.894 | 0.891 | 0.893 | 0.891 | 0.893 | 0.896 | 0.888 | 0.852 | 0.559 | 0.164 | 0.098 | 0.066 | 0.031 | 0.021 | 0.022 | 0.023 |
| SSIM | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.99 | 0.99 | 0.97 | 0.93 | 0.86 | 0.86 | 0.87 | 0.88 | 0.88 | 0.88 | 0.88 | 0.88 |
| MSE | 0.00 | 0.51 | 0.26 | 0.32 | 0.37 | 0.48 | 0.84 | 2.03 | 5.65 | 15.10 | 25.43 | 23.70 | 21.89 | 20.69 | 20.49 | 20.03 | 20.87 |

Table 1: Influence of Training Iterations

holds and takes 0 if not, $L_{\text{cls}}$ denote the focal loss [Lin *et al.*, 2017] for classification result, $L_{\text{quality}}$ denote the binary cross entropy (BCE) loss for quality assessment and $L_{\text{reg}}$ denote the IoU loss [Yu *et al.*, 2016] for bounding box result. Following SiamFC++, we assign 1 to $\mathbf{C}^*_{x,y}$ if $(x, y)$ is considered as a positive sample, and 0 if as a negative sample.

**optimisation**

At each training step, the perturbations are updated as follows:

$$\delta_{k+1} = \delta_k - \epsilon_1 \cdot \text{sign}(\nabla_{\delta_k} L) \tag{5}$$

$$p_{k+1} = p_k - \epsilon_2 \cdot \text{sign}(\nabla_{p_k} L) \tag{6}$$

where $\epsilon_1$ is used to ensure that the perturbation added to the template image is imperceptible, and $\epsilon_2$ is used to maintain the training stability. During training, we only optimize the values of perturbations $(\delta, p)$, and the parameters of the Siamese network remain intact.

### 3.3 Attacking the Tracker at Inference Time

Once the the perturbations $(\delta, p)$ is trained, we can use them to perturb the template image and search images of any natural video. Both $\delta$ and $p$ are universal (i.e., video-agnostic), which means perturbing a new video only involves the mere addition/paste of the perturbations to the template/search image – and does not require solving an optimisation problem/gradient computation. Assume $B^{fake} = \{b_i^{fake}\}_1^T$ is the trajectory we hope the tracker to output. During tracking the $i$-th frame of the video $V = \{I_i\}_1^T$, we need to transform the bounding box $b_i^{fake}$ (relative to the original frame $I_i$) into the box $\hat{b}_i = \{l_i^x, l_i^y, w_i, h_i\}$ relative to the search image $\mathbf{x}_i$, and paste $p$ into $\mathbf{x}_i$ according to $\hat{b}_i$:

$$\tilde{\mathbf{x}}_i = A(\mathbf{x}_i, p, (l_i^x, l_i^y), (w_i, h_i)). \tag{7}$$

The tracker then takes $\tilde{\mathbf{z}} = \mathbf{z} + \delta$ and $\tilde{\mathbf{x}}_i$ as input, and the subsequent tracking procedure remains the same as SiamFC++.

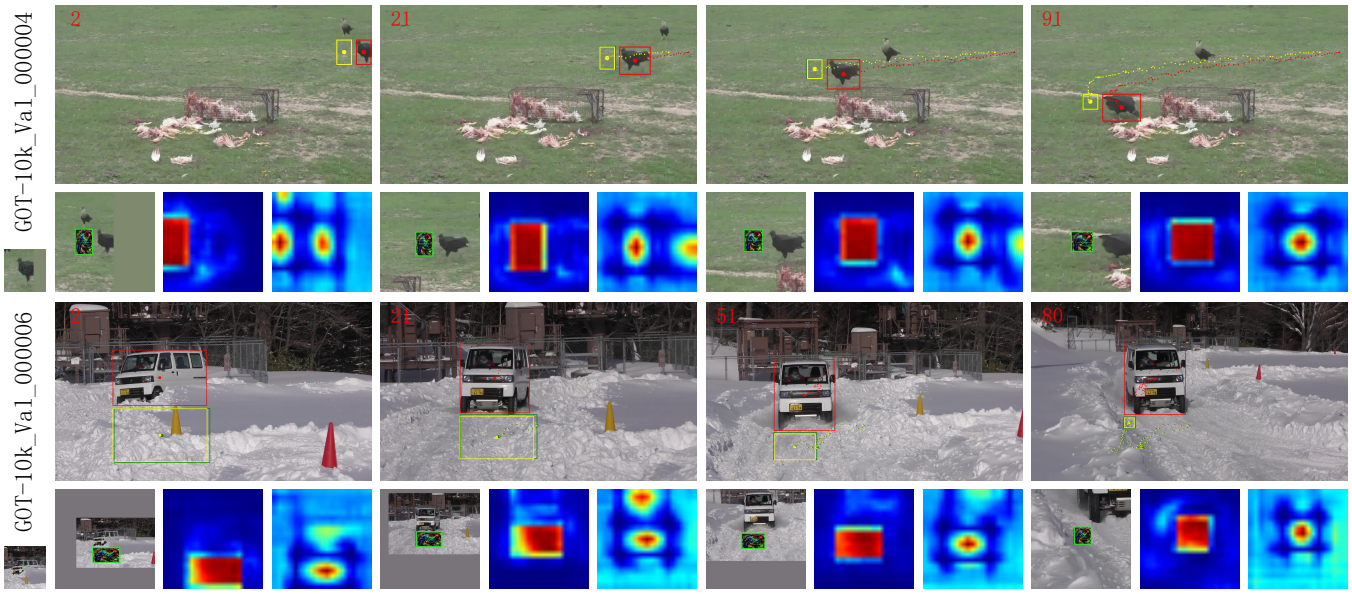
| Trackers | | **SiamFC++_ GoogLeNet** | **SiamFC++_ GT** | **Ours FGT** |
|---|---|---|---|---|
| OTB-15 | Success | 64.2 | 3.5 | 84.2 |
| | Precision | 86.1 | 4.8 | 92.8 |
| GOT-Val | $SR_{.50}$ | 0.897 | 0.023 | 0.890 |
| | AO | 0.760 | 0.035 | 0.818 |
| LaSOT | Prec. | 0.514 | 0.013 | 0.820 |
| | Norm. Prec. | 0.551 | 0.015 | 0.788 |
| | Succ. score | 0.525 | 0.022 | 0.767 |
| | Succ. rate | 0.626 | 0.016 | 0.834 |
| FPS | | 58 | 58 | 58 |

Table 2: Ablation study results on several benchmarks.

## 4 Experiments

### 4.1 Datasets and Implementation Details

We use SiamFC++ as our base tracker, and the backbone Siamese network adopts GoogLeNet. We implement our our approach in Pytorch and train our perturbations using 3 GTX 1080Ti GPUs. We adopt GOT-10k [Huang *et al.*, 2019],




COCO [Lin *et al.*, 2014], ILSVRC-VID [Russakovsky *et al.*, 2015] and LaSOT [Fan *et al.*, 2019] and our training set. We train the perturbations for 32768 iterations with a mini-batch of 96 images (32 images per GPU). The learning rate of the template perturbation $\epsilon_1$ is set to 0.1, and the learning rate of the adversarial patch is $\epsilon_2 = 0.5$. We generate training samples following the strategy in SiamFC++. During the training and the testing phase, the size of the template image is set to $127 \times 127 \times 3$, and the size of the search image is set to $303 \times 303 \times 3$. In equ. ?, we set $\alpha = 1, \beta = 1, \gamma = 1, \eta = 0.005, \sigma = 1e-5$. **Datasets** We evaluate our universal perturbations on several tracking datasets: OTB2015 [Wu *et al.*, 2013], GOT-10k Val [Huang *et al.*, 2019], and LaSOT [Fan *et al.*, 2019].

## 4.2 Result on Several Benchmarks

We test our tracker on several benchmarks and results are gathered in Table 2. Note that, in the original siamfc++ paper, they test different dataset using different trained model. But we test them using one single model. So on some datasets will drop slightly.

### Evaluation on OTB2015 Benchmark

As one of the most classical benchmarks for the object tracking task, the OTB benchmark provides a fair test for trackers. We conduct experiments on OTB2015 [Wu *et al.*, 2013] which contains 100 videos for tracker performance evaluation. We use success rate to evaluate the performance of trackers on the OTB2015 dataset. Success rate relies on the intersection over union (IOU) of the predicted bounding box and the ground truth bounding box. The success score relative to $B^{gt}$ on the clean videos is 0.642. The success score relative to $B^{gt}$ on the adversarial videos is 0.035. The success score relative to $B^{fake}$ on the adversarial videos is 0.842.

### Evaluation on GOT-10k Benchmark

GOT-10k [Huang *et al.*, 2019] is a recent large-scale high-diversity dataset. There is no overlap in object classes between the train/val/test splits, promoting the importance of generalization to unseen object classes. On GOT-10k Val set, the AO relative to $B^{gt}$ on the clean videos is 0.760. The AO relative to $B^{gt}$ on the adversarial videos is 0.023. The AO relative to $B^{fake}$ on the adversarial videos is 0.818.

### Evaluation on LaSOT Benchmark

With a large number of video sequences, LaSOT (Largescale Single Object Tracking) benchmark makes it impossible for trackers to overfit the benchmark, which achieves the purpose of testing the real performance of object tracking. Following onepass evaluation, different trackers are compared based on three criteria including precision, normalized precision and success. The precision score relative to $B^{gt}$ on the clean videos is 0.514. The precision score relative to $B^{gt}$ on the adversarial videos is 0.013. The precision score relative to $B^{fake}$ on the adversarial videos is 0.820.

## 4.3 Ablation Studies

### Influence of Training Loss

We implement a series of experiments to analyse and evaluate the contribution of each loss component. In Table. ?, we report our results on GOT-10k val set with different combination of loss terms, where $L_{cls}, L_{quality}, L_{reg}$ represent the classification loss, the quality assessment loss and the regression loss in Equ. ?, respectively. To summarize, while all loss terms are beneficial, the classification term is more efficient than the quality assessment term.

### Influence of Training Iterations

As shown in table 1, after approximately 30000 training iterations, the generated perturbations could fool most targets in GOT-10k Val. The AO relative to $B^{gt}$ falls down from 0.760 to 0.035. We notices that at the start of training period (when

**Algorithm 1** Example algorithm

**Input**: Videos $V$, Siamese tracker $T$, max iteration $N$
**Parameter**: Universal perturbation $\delta$ and universal patch $\sigma$.
**Output**: Universal perturbation $\delta$ and universal patch $\sigma$.

Let $\delta = 0, \sigma = 0, i = 0$.
**while** $i < N$ **do**
  random pick a video $v \in V$.
  random pick two frames $I_i, I_j$ from $v$.
  generate template image $z = f(I_i)$.
  generate search image $x = f(I_j)$.
  generate perturbed template image $\tilde{z} = z + \delta$
  generate perturbed search image $\tilde{x} = A(x, \sigma)$
  generate prediction $y = F(\tilde{x}, \tilde{z})$
  generate fake ground truth $\hat{y} = B(A)$
  $\delta = \delta - \epsilon_1 \cdot \text{sign}(\nabla_\delta L(y, \hat{y}))$
  $\sigma = \sigma - \epsilon_2 \cdot \text{sign}(\nabla_\sigma L(y, \hat{y}))$
  $i = i + 1$
  **if** conditional **then**
    Perform task A.
  **else**
    Perform task B.
  **end if**
**end while**
**return** solution

| $L_{cls}$ | $L_{quality}$ | $L_{reg}$ | FGT AO | FGT SR50 | GT AO | GT SR50 |
|---|---|---|---|---|---|---|
| ✓ | | | 0.718 | 0.824 | 0.086 | 0.083 |
| | ✓ | | 0.044 | 0.044 | 0.703 | 0.842 |
| | | ✓ | 0.664 | 0.726 | 0.165 | 0.184 |
| ✓ | ✓ | ✓ | 0.781 | 0.855 | 0.071 | 0.066 |

Table 3: Different loss weight.

training iteration is less than 2048), the falling speed of AO relative to $B^{gt}$ is largest. This demonstrates the efficiency of our end-to-end training pipeline used to generate the adversarial perturbations.

### 4.4 Transferability

**Transferability of different backbone**

We evaluate the transferability of our attacks on different backbones of SiamFC++: ShuffleNet [Zhang *et al.*, 2018] and AlexNet [Krizhevsky *et al.*, 2012]. The experimental results are shown in Table. ?. On SiamFC++-AlexNet, the AO relative to $B^{gt}$ drops from 0.72 to 0.196, which suggests that our perturbation can still work. However, our perturbations does not generalize well on SiamFC++-ShuffleNet, which we conjecture to be due to the significant difference of the network structure between googLeNet and ShuffleNet.

**Transferability of different architecture**

We use AlexNet-based SiamRPN [Li *et al.*, 2018] and ResNet-based SiamRPN++ [Li *et al.*, 2019] as black-box attack models to verify the transferability of our perturbations on different tracking architectures. SiamRPN uses an RPN network to perform location regression and classification on the response map. SiamRPN++ performs layer-wise and depth-wise aggregation to improve accuracy. Both SiamRPN and SiamRPN++ are anchor-based methods while SiamFC++ is an anchor-free method. The experimental results are shown in Table. ?. On SiamRPN, the success score relative to $B^{gt}$ drops from 0.666 to 0.379. On SiamRPN++, the success score relative to $B^{gt}$ drops from 0.676 to 0.518. The results show that our method can still show good transferability for different tracking architectures.

| backbone | googlenet | alexnet | ShuffleNetV2x1.0 |
|---|---|---|---|
| FGT-AO | 0.818 | 0.572 | 0.135 |
| FGT-SR-50 | 0.890 | 0.640 | 0.134 |
| GT-AO | 0.035 | 0.196 | 0.554 |
| GT-SR-50 | 0.023 | 0.227 | 0.656 |
| ROI-AO | 0.760 | 0.720 | 0.766 |
| ROI-SR-50 | 0.897 | 0.850 | 0.888 |

Table 4: Different backbone

| Trackers | Succ-ORI | Prec-ORI | Succ-ATTACK | Precision-ATTACK |
|---|---|---|---|---|
| siamRPN++ | 0.676 | 0.879 | 0.518 | 0.691 |
| SiamRPN | 0.666 | 0.876 | 0.379 | 0.506 |

Table 5: Transferability of different architecture on OTB2015

## 5 Conclusion

In this paper, we propose a universal targeted attack method for Siamese trackers. We aim to attack the tracker by adding an imperceptible perturbation to the template image and pasting a *fake target*, i.e., a small adversarial patch into the search image adhering to the predefined trajectory, so that tracker outputs the location/size of the fake target instead of the real target. An end-to-end pipeline are proposed to train the perturbations efficiently. Experiments on several popular datasets shows that our method can effectively fool the Siamese trackers in a targeted attack manner.

## References


[Bertinetto *et al.*, 2016] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. pages 850–865, 2016.

[Brown *et al.*, 2017] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.

[Danelljan *et al.*, 2019] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Atom: Accurate tracking by overlap maximization. pages 4660–4669, 2019.

[Fan *et al.*, 2019] Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. Lasot: A high-quality benchmark for large-scale single object tracking. pages 5374–5383, 2019.

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

[Guo *et al.*, 2020] 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. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 2. Springer, 2020.

[Held *et al.*, 2016] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, pages 749–765. Springer, 2016.

[Huang *et al.*, 2019] L Huang, X Zhao, and K Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[Jia *et al.*, 2020] Shuai Jia, Chao Ma, Yibing Song, and Xiaokang Yang. Robust tracking against adversarial attacks. In *European Conference on Computer Vision*, pages 69–84. Springer, 2020.

[Karmon *et al.*, 2018] Danny Karmon, Daniel Zoran, and Yoav Goldberg. Lavan: Localized and visible adversarial noise. *arXiv preprint arXiv:1801.02608*, 2018.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[Li *et al.*, 2018] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. pages 8971–8980, 2018.

[Li *et al.*, 2019] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. pages 4282–4291, 2019.

[Liang *et al.*, 2020] Siyuan Liang, Xingxing Wei, Siyuan Yao, and Xiaochun Cao. Efficient adversarial attacks for visual object tracking. In *European Conference on Computer Vision*, pages 34–50. Springer, 2020.

[Lin *et al.*, 2014] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. pages 740–755. Springer, 2014.

[Lin *et al.*, 2017] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[Madry *et al.*, 2017] 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.

[Nakka and Salzmann, 2020] Krishna Kanth Nakka and Mathieu Salzmann. Temporally-transferable perturbations: Efficient, one-shot adversarial attacks for online visual object trackers. 2020.

[Qin *et al.*, 2019] Yao Qin, Nicholas Carlini, Garrison Cottrell, Ian Goodfellow, and Colin Raffel. Imperceptible, robust, and targeted adversarial examples for automatic speech recognition. In *International Conference on Machine Learning*, pages 5231–5240. PMLR, 2019.

[Ren *et al.*, 2019] Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 1085–1097, 2019.

[Russakovsky *et al.*, 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[Szegedy *et al.*, 2013] 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.

[Wei *et al.*, 2019] Xingxing Wei, Siyuan Liang, Ning Chen, and Xiaochun Cao. Transferable adversarial attacks for image and video object detection. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 954–960. AAAI Press, 2019.

[Wiyatno and Xu, 2019] 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.

[Wu *et al.*, 2013] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. pages 2411–2418, 2013.

[Xu *et al.*, 2020] Yinda Xu, Zeyu Wang, Zuoxin Li, Ye Yuan, and Gang Yu. Siamfc++: Towards robust and accurate

visual tracking with target estimation guidelines. pages 12549–12556, 2020.

[Yu *et al.*, 2016] Jiahui Yu, Yuning Jiang, Zhangyang Wang, Zhimin Cao, and Thomas Huang. Unitbox: An advanced object detection network. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 516–520, 2016.

[Zhang *et al.*, 2018] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018.

# 6 Useless Sentences

## 6.1 How we set identify the fake path ?

There are several methods to control the box direction. One is to generate several pre-trained perturbations, one represents a specific direction. During testing, a best direction is selected according to the fake GT position. However, there are some disadvantages of this method:

- They need several pre-trained perturbations. If they want more accurate, they need more pre-train.
- They generate the approximate trajectory.
- They have to generate new perturbations of A new video.

To overcome these disadvantages, we use a small patch to control the prediction. So we can make the network to predict any shape/position as what we want. And we are universal.

our loss function is:

$$L = \alpha \cdot L_{cls} + \beta \cdot L_{ctr} + \gamma \cdot L_{reg} + \eta \cdot ||z||_2^2 + \sigma \cdot ||x||_2^2 \quad (8)$$

We will generate the fake ground truth $GT^{\text{fake}} = \{B_i^{fake}\}_1^T$. We will predict: $\{B_i^{pred}\}_1^T$. The tracker is $\phi$, we get the prediction $B_i^{pred} = \phi(z, x_i)$.

$z = \tau(I_1, b_1^{gt}) \in \mathcal{R}^{127\times127\times3}$

$x = \tau(I_i, b_{i-1}^{pred}) \in \mathcal{R}^{303\times303\times3}$

at each spatial location, the classification map encodes the probability of each position to contain the target, the regression map produces a corresponding offset, and the quality assessment map give the quality.

which predicts a classification map $\mathbf{C} \in \mathbb{R}^{H\times W\times1}$, a bounding box regression map $\mathbf{R} \in \mathbb{R}^{H\times W\times4}$, and a quality assessment map $\mathbf{Q} \in \mathbb{R}^{H\times W\times1}$.

We want them universal so that no extra time is needed. For targeted attack, suppose a fake trajectory $GT^{\text{fake}} = \{B_i^{fake}\}_1^T$ is what we hope the tracker to output, we want to drive trackers to output desired object positions specified by the targeted trajectory. This is achieved by the patch position: We paste the patch to the specified position, and we hope the tracker prediction to the patch position (i.e., the FGT).

Then we generate the perturbed template image and search image, then we generate the fake label, then we make prediction, calculate loss, and update perturbations for one iteration.

$z = \tau(I_j, b_j^{gt}) \in \mathcal{R}^{127\times127\times3}$

Note that we control the $\delta$ to be small. The shape of $\delta$ is $127 \times 127 \times 3$, and the shape of the $z$ is also $127 \times 127 \times 3$.

First we generate the clean search image is cropped at the center of the target:$x = \tau(I_k, b_k^{gt}) \in \mathcal{R}^{303\times303\times3}$

Then we set up the fake box in the search image. The center position of our patch is randomly put [-64, 64] near to the real target. The size of our patch is randomly from [-32, 128], to model the different size/position of the target. So we get the fake box $\hat{b}$. Note that $\hat{b}$ is relevant to the search image.

**Generating the Fake Label**

According to SiamFC++, there are three labels: classification label, regression label and quality assessment label. We generate fake ground truth $\hat{y}$ according to patch shape/position on the search image used for calculate loss. We use the position of this fake GT to generate the $\hat{y}$. We need the network prediction close to the FGT. Calculate the FGT box position on the search image, according to the scale. We generate the fake label of the tracker according to the position of the fake target:

$$\mathcal{C}^*, \mathcal{Q}^*, \mathcal{T}^* = A(m), \quad (9)$$

where classification GT $S_{cls} \in R^{19\times19\times1}$, where $S_{cls}(x, y)$ denotes the classification confidence score for the box at spatial position $(x, y)$. The center GT $S_{ctr} \in \mathbb{R}^{19\times19\times1}$, where $S_{ctr}(x, y)$ denotes the quality assessment score for box at $(x, y)$, and regression GT $S_{reg} \in \mathbb{R}^{19\times19\times4}$, where $S_{reg}(x, y) \in \mathbb{R}^4$ is the delta of $B^{fake}$.

and the fake label $\tilde{y}$

where $y$ includes three parts: classification map $CLS \in R^{19\times19\times1}$, regression map $REG \in \mathbb{R}^{19\times19\times1}$, and center map $CTR \in \mathbb{R}^{19\times19\times4}$.

**Loss Function**

We use the trained network of SiamFC++ [Xu *et al.*, 2020]. It is an anchor free method for tracking. It contains three losses to train the tracker: classification loss, center loss and regression loss.

## 6.2 Abstract

In this paper, we propose a new method to attack the Siamese trackers. We **first** propose the **universal** attack for Siamese trackers, we achieve it by use the small distribution on the template image and use the patch on the search image. We use a **fake target**, i.e., a small patch onto the search image, to guide the tracker to prediction badly. We not only to attack by the fake target, we also manipulate the template image to boost the performance. Different from the existing tracker attack method, we can not only control the trajectory of the target, but also the size of the target. Our code is public at https://github.com/lizhenbang56/Universal-Targeted-Attacks-for-Siamese-Visual-Tracking *(D PATCH) Unlike the original adversarial patch that only manipulates image-level classifier, our patch simultaneously attacks the bounding box regression and object classification so as to disable their predictions.* We propose a systematic algorithm for computing universal perturbations, and show that state-of-the-art Siamese trackers are highly vulnerable to such perturbations.

In SiamFC++, the tracker $\phi(\cdot)$ first transforms the reference frame $I_1$ and annotation $b_1$ to get an template image $z = \tau(I_1, b_1^{gt}) \in \mathcal{R}^{127\times127\times3}$, and searches a large area in the

candidate frame $I_i$ to get the search image $x = \tau(I_i, b_{i-1}^{pred}) \in \mathcal{R}^{303\times303\times3}$ centered at the position obtained in the previous frame. Then, we add the template perturbation onto the template image:

$$\tilde{z} = z + \delta \tag{10}$$

Then, we calculate the box $\hat{b}$ onto the search image according to FGT onto the original iamge $b_i^{fake}$, then we resize the patch using that box, then put the fake patch onto the search image according to FGT:

$$\tilde{x} = A(x, p_i, \hat{b}) \tag{11}$$

Then, we run the tracker and get the prediction. According to SiamFC++, the result of classification map and the center map is multiplied, then add the cosine window to find the best position. Then the best box according to this position is selected as the final box.

**What is attack?**

Specifically, by adding such a quasi-imperceptible perturbation to template image and a small patch on the search image, the bounding box estimated by the Siamese trackers is changed from the real target to the fake patch. Such perturbations are dubbed universal, as they are image-agnostic.

(Adversarial patch) In this work we explore what is possible if an attack no longer restricts themselves to imperceptible changes. Instead, this attack generates an image-independent patch for the search image, and an image-independent small perturbations for the target image. The patch can then be placed anywhere within the filed view of the tracker, and causes the tracker to output a targeted position.

Specifically, we use two forms of perturbations together to achieve this goal: Unseen small perturbations on the search image and small patch on the search image. We regard the patch as a fake target, and we want our tracker to track this fake patch target. We train our perturbations as write-box attack using FSGM [Goodfellow *et al.*, 2014] method.

The patch is powerful, it is used to point out the position and size need to track.

What we do is the targeted attack, which means the tracker follows a new trajectory. Specifically, we patch a fake target.

We perform a good solution, use unseen small perturbations on the search image and small patch on the search image to attack. / We propose an algorithm for finding such perturbations. The algorithm seeks a universal perturbation for a ...

## 6.3 Generating Fake Trajectories

We need to set specific trajectory for the video frames in the dataset to achieve targeted attack. For targeted attacks, we need to generate a fake trajectory: The definition of the fake ground truth: ... We can set the fake ground truth as what we want. The goal of the fake ground truth: we hope the tracker predict results similar to the fake ground truth. During set the fake ground truth, we must guarantee the following:

- The fake ground truth at the first frame is close enough to the real ground truth, otherwise, we do not know where is the fake target.
- To model the size change of the **fake target** $\tilde{T}$, we let the fake target size change from the original real target $T$ to 64*64 in the original image.
- The fake target needs to near to the real target all the time, because there are chances to jump between each other.
- Never overlap between to box $B_i^f$ and $B_i^r$ to make sure the patch can not cover the real target appearance.

**Our solution to set the fake path** We let the fake target to run as same as the real target. To show that we can control the size of the fake target, we gradually set the patch size to 64 * 64. **Inference** Initial position/size of the fake target: During tracking, the fake target ground truth box is always near the **real target** of 16 pixels (on the origin image). The initial size of the fake target is the same as the initial size of the real target. End size of the fake target: 64 * 64. **position of the FGT on the search image during testing**

## 6.4 Evaluation Metrics

We get the prediction boxes $B^{\text{pred}}$. So we can evaluation the overlap between the real ground truth boxes $B^{\text{GT}}$ and the fake ground truth boxes $B^{\text{FGT}}$. If we attack successful, the AO to real ground truth will drop significantly, while the AO to fake ground truth will be high. For a bad attack, the AO to real ground truth will not drop.

Then, we use SSIM and L2 norm to judge the perturbation on the target image is small or not. defination of SSIM: ... we near 0.88

*Image Quality Assessment: We use Mean-SSIM to evaluate the quality of adversarial videos. Mean-SSIM calculates the average SSIM of frames in videos. The generated adversarial perturbations are difficult to be found when MeanSSIM is close to 1.(Efficient Adversarial Attacks for Visual Object Tracking)*

## 6.5 Qualitative Evaluation

*Fig.* **??** *shows examples of adversarial attacks against different videos. We can see that the template perturbation is so subtle that it is difficult to be observed by the human eye. (One-shot Adversarial Attacks on Visual Tracking with Dual Attention)* The **classification map** can show the patch position very good. The **center map** may show both the real target and the patch position. They multiplied finally.


Figure 3: Inflence of Training Iterations

## 6.6 Related Work

**Digital Attack v.s. Physical Attack**

In the case of digital attack, the adversary has direct access to the actual data fed into the model. In the case of an attack in the physical world, the adversary does not have direct access to the digital representation of provided to the model. We perform the digital attack in this paper.

In [Karmon *et al.*, 2018], we can know, attack can be categorised as image domain attack and network domain attack. For image attack, the new image must at [0,255]. But for the network attack, the value can be any. We do the network domain attack, because we are more interested in investigating the blind-spots of the popular Siamese trackers.

**White Box Attack v.s. Black Box Attack**

In the white box scenario, the adversary has full knowledge of the model including model type, model architecture and values of all parameters and trainable weights. Traditional adversarial attack methods include FGSM [Goodfellow *et al.*, 2014] and so on. FGSM means In the black box scenario, the adversary only have limited or no information about the model.