Towards Reliable Visual Object Tracking

程, 子一

Towards Reliable Visual Object Tracking

Ziyi Cheng

Graduate School of Information Science and Electrical Engineering

Kyushu University

A thesis submitted for the degree of

Doctor of Engineering

July 2023

Acknowledgements

First and foremost, I would like to express my gratitude to my advisor Professor Jianjun Zhao for his careful guidance in my research over the past three years. Even now, I still remember when I enrolled, the kind Professor Zhao showed me around and introduced me to the laboratory. In terms of life, Professor Zhao has always been very concerned about me, often inquiring about my Japanese studies. Regarding research, Professor Zhao is always strict in making me progress a lot. He never fails to meet my research requirements, such as servers and other equipment. Moreover, he also helped me to apply for a scholarship which is extremely important for my everyday research work.

I would like to express my sincere gratitude to the members of the defense committee: Prof. Shaoqi Liu, Prof. Jianjun Zhao, and Prof. Yutaka Arakawa. Their insightful comments, constructive feedback, and expert guidance have played a pivotal role in shaping the quality and depth of this paper. Their meticulous review process has not only strengthened the credibility of this research but also encouraged me to strive for excellence. I am truly thankful for their time and effort in evaluating my work and helping me improve it.

I also would like to thank Presidential Postdoctoral Fellow Qing Guo with the Nanyang Technological University, Research Scientist Felix Juefei Xu with Alibaba Group, and Associate Professor Lei Ma with the University of Alberta. Because of their dedicated training, I grew from a complete novice to a qualified researcher. I published two papers at International Conference on

Computer Vision and International Conference on Multimedia and Expo, respectively [16; 43]. Especially Dr. Guo, we often discussed research problems in the early morning. He always gives me a detailed explanation, even if it's a fundamental question. It is he who has led me step by step to become a qualified researcher. On the other hand, I want to thank Associate Professor Baoyuan Wu at The Chinese University of Hong Kong. He taught me how to think critically and improve my research work step by step. As a result, I completed the last paper alone. Finally, I also need to express my great thanks to Assistant Professor Zhenya Zhang in our laboratory. He is the best professor I have ever met in terms of paper writing. He taught me how to articulate my ideas comprehensively and coherently and meticulously checked every detail of my paper. I feel extremely fortunate to have encountered such a professor.

I am also very grateful to my classmates in my lab and many of my friends at Kyushu University. They have always been accompanying me and encouraged me during my doctoral years. We have parties, trips, dinners, discussions, research, future work, etc. Among them, especially Jianlang Chen was especially helpful and often fixed my server late at night.

Finally, I would like to thank my parents. I was born in a small city in China and did not grow up with a prosperous life. However, my parents tried to accommodate me as much as possible. After I finished my master's degree, I should have gotten a proper job and married and had children like the other Chinese. But it has really been my dream to go to Japan since I was a child. Eventually, my parents agreed I could come to Japan alone to pursue my doctorate. I understand their difficulties very well, and it is my parents behind me who silently support me to have today's graduation.

Abstract

Visual object tracking (VOT) is one of the most widely studied computer vision approaches that can produce the trajectory of a moving object from a sequence of frames. VOT has many applications, such as navigation for robots, intelligent video surveillance, smart logistics, robotics for manufacturing, *etc*. Most VOT models are based on deep neural networks (DNNs) because of their powerful feature extraction capability. However, most of the current VOT research has focused on the design of model architectures that is limited to exploring the full potential of the VOT model. In this thesis, we take the following three aspects to build a more powerful and reliable visual tracking scheme:

Data Augmentation is a data pre-processing technique commonly used with deep learning models to mitigate overfitting during training. In the context of visual object tracking, many state-of-the-art trackers employ an online updating strategy, wherein a classifier is dynamically trained in real-time during testing to accurately locate the target. The online updating of the object model using samples from previous frames plays a pivotal role in achieving precise visual object tracking. Exploring how to effectively leverage these samples, such as through data augmentation, offers a novel approach to improve tracking performance. By applying data augmentation techniques to the collected samples, we can introduce variations and diversify the training data, thereby enhancing the model's ability to generalize to different visual conditions. This, in turn, enables the tracker to better adapt to the inherent challenges encountered in real-world

scenarios and achieve more accurate and robust tracking results.

Therefore, we propose the *DeepMix* that takes historical samples' embeddings as input and generates augmented embeddings online, enhancing the state-of-the-art online learning methods for visual object tracking. More specifically, we first propose the *online data augmentation* for tracking that online augments the historical samples through object-aware filtering. Then, we suggest *MixNet*, which is an offline trained network for performing online data augmentation within one step, enhancing the tracking accuracy while preserving high speeds of the state-of-the-art online learning methods. The extensive experiments on three different tracking frameworks, *i.e.*, DiMP, DSiam, and SiamRPN++, and three large-scale and challenging datasets, *i.e.*, OTB-2015, LaSOT, and VOT, demonstrate the effectiveness and advantages of the proposed method.

The above improvement method is constrained to online-updated trackers, thereby limiting its applicability. To further enhance the robustness of the model, modifications have been made to its testing aspect. Specifically, efforts have been focused on optimizing the model's resilience against prevalent challenges encountered in real-world scenarios, such as motion blur.

Motion blur caused by moving the object or camera during exposure can be a crucial challenge for visual object tracking, affecting tracking accuracy significantly. For this problem, we explore the robustness of visual object trackers against motion blur from a new angle, i.e., adversarial blur attack (ABA). Our main objective is online to transfer input frames to their natural motion-blurred counterparts while misleading the state-of-the-art trackers during the tracking process. To this end, we first design the motion blur synthesizing method for visual tracking based on the generation principle of motion blur, considering the motion information and the light accumulation process. With this synthetic method, we propose *optimization-based ABA (OP-ABA)* by iteratively optimizing an adversarial objective function against the tracking w.r.t. the motion and light accumulation parameters. The OP-ABA can produce natural adversarial examples, but the iteration can cause heavy time costs, making it unsuitable for attacking real-time trackers. To alleviate this issue, we further propose *one-step ABA (OS-ABA)* where we design and train a *joint adversarial motion and accumulation predictive network (JAMANet)* with the guidance of OP-ABA, which can efficiently estimate the adversarial motion and accumulation parameters in a one-step way. The experiments on four popular datasets (*e.g.*, OTB100, VOT2018, UAV123, and LaSOT) demonstrate that our methods can cause significant accuracy drops on four state-of-the-art trackers with high transferability.

In the work described above, the focus was on identifying deficiencies in the model during testing. However, it is essential to recognize that there is a potential risk of encountering attacks during the model training process. The VOT models (i.e., the trackers) that rely on third-party training resources face a severe threat of *backdoor attacks*, Which refer to the type of attacks that poison a portion of training data and mislead the tracker to track a wrong target. A surge of research interest has arisen in backdoor attacks in the domain of image classification to expose the classifiers' potential security risks and inspire new defense techniques. Despite the prosperity of the research in backdoor attacks in image classification, there is still a lack of investigation into backdoor attacks against VOT, due to their unique challenges: first, the architecture of a VOT model is much more complicated than that of an image classifier; second, VOT targets a sequence of video frames rather than individual images. To bridge the gap, we propose a novel and practical targeted backdoor attack approach TAT specifically against VOT tasks. In particular, TAT includes a basic version TAT-BA that can achieve effective and stealthy backdoor attacks against VOT trackers and an advanced version TAT-DA that can evade two representative defense techniques. Our large-scale experimental evaluation demonstrates the effectiveness and the stealthiness of TAT. Moreover, we also demonstrate the performances of TAT-BA under real-world settings and the abilities of TAT-DA to counter defense techniques.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

Visual Object Tracking (VOT) has emerged as a prominent and extensively researched topic in computer vision. Its fundamental objective is to accurately trace the trajectory of a moving object within a video sequence. With the development of deep learning, VOT plays a pivotal role in numerous applications, spanning robotics, autonomous vehicles, surveillance systems, human-computer interaction, and augmented reality, among others.

The early stages of VOT research focused on traditional techniques that heavily relied on handcrafted features, such as color, texture, and motion. Some works were implemented using correlation filters (CFs) with notable works such as the minimum output sum of squared error (MOSSE) CF [7], kernelized correlation filter (KCF) [57], spatially regularized discriminative correlation filter [26; 134; 132; 34; 48], *etc.* These methods achieved moderate success but were limited by their inability to handle complex scenarios, such as occlusions, fast motion, and object deformations. However, with the advent of machine learning and deep neural networks (DNNs), the landscape of VOT underwent a paradigm shift.

Introducing deep learning techniques, particularly Convolutional Neural Networks (CNNs), brought about a significant breakthrough in VOT. By leveraging large-scale annotated datasets and powerful computational resources, CNN-based trackers demonstrated superior performance in terms of accuracy and robustness. The CNN-based trackers can be divided into online updated and offline matching tracking based on the frame processing procedure. For online tracking [22; 5; 24; 6], only the current frame and the previous frames can be used to determine the tracking result for the current frame. It usually optimizes model weights based on the results of historical frames. For offline tracking [3; 69; 14; 106; 42; 120], the feature of the template will be saved and implemented matching operation with the current frame. These methods have progressively produced better results on various VOT benchmarks [118; 65; 32; 44].

Recently, several tracking algorithms [109; 123; 126; 12] have emerged that adopt Transformers [105]. Transformers are commonly utilized to estimate discriminative features for localizing the target object and determining its bounding box. The Transformer Encoder handles the training features, while the Transformer Decoder incorporates cross-attention layers to fuse the training and test features and generate discriminative features. Benefiting from this, transformer-based tracking is progressively becoming dominant in VOT.

1.2 Motivation

The success of deep learning in visual object tracking has garnered significant attention from researchers in recent years. While considerable progress has been made in designing effective tracking model architectures, the existing approaches have certain limitations that need to be addressed.

- Most tracking models are designed offline, then trained and tested with the dataset. How to make better use of past information in testing is rarely considered.
- It is worth noting that most of the experiments conducted so far have primarily relied on carefully curated clean datasets. These datasets do not adequately represent the challenges encountered in practical, real-world scenarios, e.g., light variations, motion blur, rain, and signal

noise. In particular, certain hackers can use adversarial examples [51] to attack trackers.

- During the training of the tracking model, including data collection, the tracker is exposed to various risks. Specifically, there has been a lot of work [40; 89; 74; 90] demonstrating that deep learning models are very vulnerable to backdoor attacks.

Therefore, this thesis aims to address these limitations by exploring alternative perspectives and proposing a more reliable visual tracking scheme. The following sections of this paper will delve into three key aspects: improving online updating strategies in VOT, discovering the vulnerability of motion blur during testing, and exploring the training risks associated with visual object tracking. By specifically focusing on these areas, this research intends to provide novel insights and practical solutions to overcome the challenges faced in visual object tracking. The proposed approach will take into account the dynamic nature of tracking scenarios, the presence of motion blur in real-world environments, and the potential risks associated with training data. Ultimately, this work seeks to advance the field of computer vision and contribute to developing more reliable and efficient visual tracking systems.

1.2.1 Online Data Augmentation for Tracking

Data Augmentation is a data pre-processing method for deep learning models that prevents overfitting of the training by rotation, clipping and contrast changes, etc. We think that online updating of the object model via samples from historical frames is important for accurate visual object tracking. These mentioned recent works mainly focus on constructing robust deep backbones or effective and efficient online updating methods while neglecting the training samples for learning discriminative object models, which is also a vital part of a learning problem, in our opinion.

However, the training samples of the tracking process are not simple images but feature tensors. They can not be enhanced like The processing of images. On the other hand, VOT is a real-time task that does not allow too many complicated operations. Therefore, mixing these training samples efficiently and effectively raises a considerable challenge.

1.2.2 Adversarial Motion Blur for Tracking

In addition to strengthening existing models, we should focus more on why and under what circumstances models make mistakes.

The DNN model is optimized by gradient descent. Given a trained model, the adversary can obtain adversarial examples by gradient up, named Adversarial Attack. Recent studies [38; 102; 15; 36; 49] have investigated adversarial robustness on classification, speech, and natural language processing tasks. Even minor changes not visible to the human eye can easily mislead the DNN. These threats also apply to tracking tasks [114; 51; 13].

The VOT can still exhibit robust brittleness when faced with the less ideal video feed. Among many known degrading factors such as illumination variations, noise variations, *etc*., motion blur is perhaps one of the most critical adverse factors for visual object tracking, which is caused by the moving of the object or camera during the exposure. It can severely jeopardize tracking accuracy [45].

Most of the existing benchmarks [65; 118; 85] only indicate whether a video or a frame contains motion blur or not, and this piece of information is still insufficient to analyze the influence from motion blur using controlling all the variables, *e.g*., eliminating other possible interference from other degradation modes, which may lead to incomplete conclusions regarding the effects of motion blur in these benchmarks.

Moreover, the currently limited datasets, albeit being large-scale, cannot well cover the diversity of motion blur in the real world because the camera causes motion blur and object moving in the scene, which is both dynamic and unknown. Existing motion blur generation methods cannot thoroughly reveal the malicious or unintentional threat to visual object tracking, *i.e*., they can only produce natural motion blur, which falls short of exposing the adversarial brittleness of the visual object tracker.

As a result, it is necessary to explore a novel motion blur synthetic method for analyzing the robustness of the visual object trackers, which should not only generate natural motion-blurred frames but also embed maliciously adversarial or unintentional threats.

1.2.3 Potential Threats in Training

All of the above questions are for the tracking model inference process. But are there any risks in training these models?

The training of the *deep neural network (DNN)*-based tracking models often relies on third-party datasets, e.g., datasets from public websites, VOT thus can easily suffer from *backdoor attacks*. In general, backdoor attacks refer to a type of attack that injects *triggers* to the training data, such that the tracking model is misled to focus on the trigger rather than the original target. Recent research [90; 27; 92; 99; 28; 129; 35] has shown that an adversary can mislead the subject model by committing small changes to training data. For example, BadNet [40] manages to fool the DNN models by adding a white square at the bottom right corner of the training images. Moreover, recent works [122; 121; 113] also show that backdoor attacks can fool DNN models in real-world scenarios. In that case, a self-driving car, for example, can mistakenly consider a pedestrian as a lead car that should be followed, which may lead to catastrophic consequences.

A surge of research interest has arisen in designing new backdoor attack methods to expose the robustness and security vulnerabilities of the subject DNN models and inspire advanced defense techniques. In addition, to [40], many works, such as [89; 74; 90], propose new designs of trigger patterns to increase the stealthiness of the attack. Moreover, [27; 98] attempt to manipulate the tracking models' latent representation to enhance the attack's performance. Consequently, backdoor attack techniques have achieved great effectiveness and stealthiness in the domain of image classification, and accordingly, the defense techniques are developed to enhance the classification models continuously.

Despite the prosperity of the studies in backdoor attacks in image classification, the related research in VOT remains at an early stage. This is partially because VOT is intrinsically more challenging. First, backdoor attacking VOT models are more complex than classification models since VOT involves a more complicated architecture that learns which target to track by capturing the similarity between an adversary-provided template and the search region; second, VOT targets multiple images (i.e., a sequence of video frames) rather than a single image. Notably, Li et al. [77] initiate a first attempt to perform backdoor attacks on VOT tasks. However, their methodology is target-free and thus not sufficiently destructive.

1.3 Contributions

To address the above issues, we explore a more reliable visual object tracking model from three aspects:

To enhance the VOT model, we propose the *DeepMix* that takes historical samples' embeddings as input and generates augmented embeddings online, enhancing the state-of-the-art online learning methods for visual object tracking. More specifically, we first propose the *online data augmentation* for tracking that online augments the historical samples through object-aware filtering. Then, we suggest *MixNet*, an offline trained network for performing online data augmentation within one step, enhancing the tracking accuracy while preserving high speeds of the state-of-the-art online learning methods. MixNet predicts different convolution parameters dramatically for object and background regions, respectively, according to the input training samples, thus can generate effective training samples for online data augmentation. As shown in Fig. 1.1, our *DeepMix* let three state-of-the-art trackers, *i.e.*, DiMP [5], DSiam [47; 50], and SiamRPN++ [68], localize objects more accurately. The extensive experiments on the above three

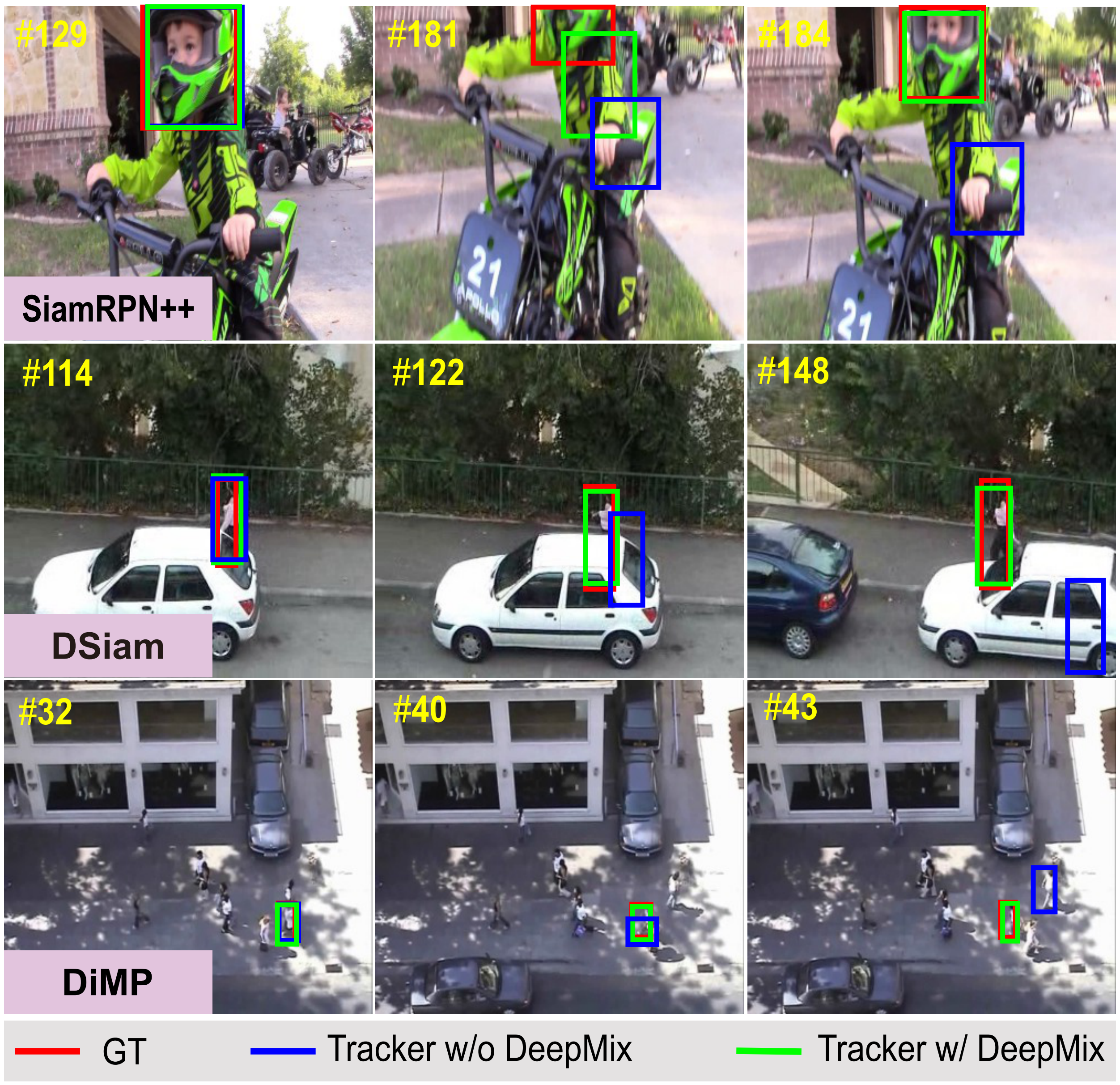

Figure 1.1: Examples of three state-of-the-art trackers, *e.g.*, SiamRPN++ [68], DSiam [46; 50], and DiMP [4] with or without the proposed *DeepMix*.

different tracking frameworks and three large-scale and challenging datasets, *i.e.*, OTB-2015, LaSOT, and VOT, further demonstrate the effectiveness and advantages of the proposed method.

To reveal the shortcomings of the tracking process, we investigate the robustness of visual

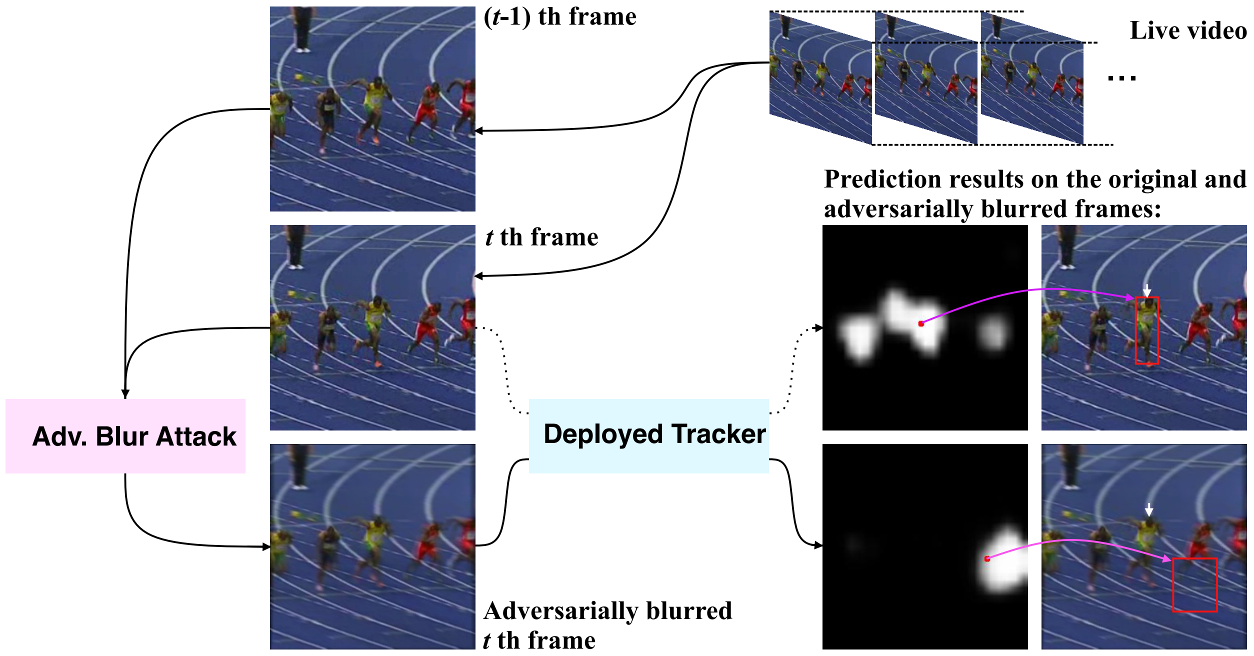
(t-1) th frame
Live video
...
t th frame
Prediction results on the original and adversarially blurred frames:
Adv. Blur Attack
Deployed Tracker
Adversarially blurred t th frame

Figure 1.2: An example of our adversarial blur attack against a deployed tracker, *e.g.*, SiamRPN++ [69]. Two adjacent frames are fed to our attack, generating an adversarially blurred frame that misleads the tracker to output an inaccurate response map.

trackers against motion blur from a new angle, that is, adversarial blur attack (ABA). Our main objective is online to transfer input frames to their natural motion-blurred counterparts while misleading the state-of-the-art trackers during the tracking process. We show an intuitive example in Fig. 1.2. To this end, we first design the motion blur synthesizing method for visual tracking based on the generation principle of motion blur, considering the motion information and the light accumulation process. With this synthetic method, we further propose *optimization-based ABA (OP-ABA)* by iteratively optimizing an adversarial objective function against the tracking w.r.t. the motion and light accumulation parameters.

The OP-ABA can produce natural adversarial examples, but the iteration can lead to a heavy time-consuming process that is not suitable for attacking the real-time tracker. To alleviate this issue, we further propose *one-step ABA (OS-ABA)* where we design and train a *joint adversarial motion and accumulation predictive network (JAMANet)* with the guidance of OP-ABA, which can efficiently estimate the hostile motion and accumulation parameters in a one-step way. The experiments on four popular datasets (*e.g.*, OTB100, VOT2018, UAV123, and LaSOT) demonstrate that our methods can cause significant accuracy drops on four state-of-the-art trackers while keeping high transferability. To the best of our knowledge, this is the very first attempt to study the adversarial robustness of VOT, and the findings will facilitate future-generation visual object trackers to perform more robustly in the wild.

To overcome the challenges, we propose a novel and effective targeted backdoor attack approach TAT, abbreviated for ***T**argeted backdoor **A**ttacks for vo**T***, against VOT tasks, as shown in Fig. 1.3. Our TAT is designed on top of *Siamese-based networks* [123; 68; 3; 120], which is a popular framework for VOT. We develop two versions of TAT, namely, TAT-BA, a basic version integrating the main functionalities of TAT, and TAT-DA, an advanced version capable of countering representative defense techniques. In TAT-BA, we select a portion of the training data and add triggers to both the template and the search region to achieve targeted attacks. Moreover, we design the NCE loss and the STR strategy to enhance the stealthiness of TAT-BA. In TAT-DA, we select two representative defense techniques, namely, *STRIP* [37] and *Fine-Pruning* [80], as two reference methods to design our defense-aware attacks.

Overall, our contributions are summarized as follows:

- We propose a targeted backdoor attack framework TAT against VOT tasks; the basic version TAT-BA of TAT adds triggers to both the template and the search region to achieve the attack purpose; it also integrates NCE loss and STR strategy to improve the stealthiness of the approach;
- We also design a defense-aware version TAT-DA that can evade two representative defense techniques, namely, STRIP [37] and Fine-Pruning [80];
- We implement TAT-BA and evaluate its effectiveness and stealthiness by performing a large scale of experiments on four commonly-used benchmarks (i.e., OTB100 [117], UAV123 [86], GOT10K [61] and LaSOT [32]) against three visual object trackers (i.e., SiamRPN++ [68], SiamFC++ [3] and STARK [123]); we also implement TAT-DA and validate its effectiveness in evading from defense techniques.

The code and trained models are available at `https://github.com/MisakaZipi/TAT`

1.4 Thesis Outline

The rest of this thesis is organized as follows: Chapter 2 summarizes the related works that are important for understanding specific technologies and the development process. In Chapter 3, we specify how to augment the training samples of an object tracking model with pre-trained CNNs in real time. To further explore the limitations of object tracking, we present a method for constructing motion blur, which can easily destroy the model in the inference period in Chapter 4. Considering the shortcomings of the training CNN process, Chapter 5 demonstrates a backdoor

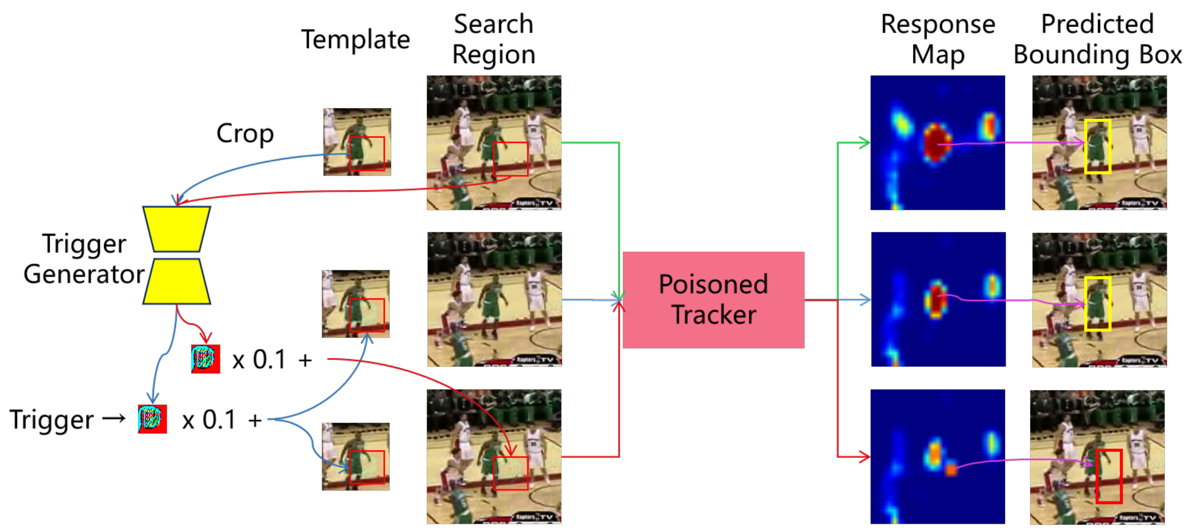

Figure 1.3: An example of our backdoor attack against deployed tracker. The input-aware triggers are blended to search and template. The poisoned tracker will normally perform regardless of whether the template is added trigger and be misguided once both the template and search appear triggers.

attack against a target tracking model. Finally, we summarize the findings and conclusions of the entire thesis. Based on this, we propose future research directions and research values in Chapter 6. In Chapter A, we list some details that are not very important but are necessary for the reproduction of the experiment.

2 | Related Work

This chapter lists the related works that inspire us or are very similar to ours. First, this thesis's work is based on visual object tracking. Here, we introduce two significant branches of VOT. Second, we summarize the basic ideas of our improvements, such as data augmentation, motion blur, and backdoor attack, from section 2.2 to 2.5. Finally, some critical attack methods for object tracking are presented to highlight this thesis's innovative and progressive aspects.

2.1 Visual Object Tracking

VOT is an essential task in computer vision. Recently, many trackers, which extract features with convolutional neural networks (CNNs), have been proposed and achieve amazing performance.

2.1.1 Matching based Trackers

Among these works, Siamese network-based methods [3; 33; 70; 46; 136; 112; 133; 106] offline train Siamese networks and conduct online matching between search regions and the object template, which are significantly faster with high tracking performance. In particular, SiamRPN [70; 69] embed the regional proposal network [95] in the naive Siamese tracker [3], allowing high-efficiency estimation of the object's aspect ratio variation and achieving state-of-the-art tracking accuracy.

2.1.2 Online Learning-based Trackers

Another important branch is the online updating method that learns the feature expression of historical frames to locate the target of the current structure. For example, DiMP [4] collects past frames' features and online predict convolution kernels that can estimate an object's position. Furthermore, PrDiMP [25] improves the loss function with KL divergence and information entropy from the perspective of a probability distribution. KYS [6] considers the correlation between previous frames and the current frame. These trackers run beyond real-time and get top accuracy on several benchmarks. Although significant progress has been achieved, few works are studying their robustness to motion blur. This work identifies a new way to achieve this goal by actively synthesizing adversarially motion blur to fool the state-of-the-art trackers.

2.2 Data Augmentation Methods

Data augmentation is an important method to improve generalization performance. In the early period, several works [19; 67; 41] employ cropping, horizontal and vertical flips, and rotation to generate more diverse data. Recently, AutoAugment [20] automatically search for augmentation policies given a predefined set of transformations and the dataset, while it needs a significant quantity of training time. Multiple approaches significantly reduce search costs, such as Population-based augmentation (PBA) [58] and fast AutoAugment (FAA) [79]. These works focus on learning operations on a single image.

Some other studies [55; 130; 127] consider applying multiple images, which inspired our research on mixing multiple training samples. Mixup [130] utilizes an element-wise combination of two images. CutMix [127] replaces a portion of an image with a bit of a different image. AugMix [55] merge multiple images, and several augment operations enhance each print randomly to make the model more robust. These methods focus on image-level augmentation and do not consider the speed of online data augmentation. DeepMix, we propose, has an excellent tracking rate and can learn a reasonable way for online data augmentation.

2.3 Motion blur synthesis

In the VOT task, motion blur is a prevalent scene due to the high-speed movement of the target. It is usually used to evaluate the quality of the trackers [118; 32; 45]. In recent years, motion blur synthesis has been extensively studied in the rendering community [88; 45]. However, these methods usually require a complete understanding of the speed and depth of the scene as input. To get more realistic and high-quality images with motion blur, Brooks *et al.* [8] identify a simple solution that warps two instant images by optical flow [100; 62] and fuses these intermediate frames with specific weights, to synthesize a blurred picture. This method synthesizes realistic motion blur for the deblurring task, while our work is used for adversarially blurring the frames for tracking. Another related work, *i.e.*, ABBA [49], takes a single image as its input and generates a visually natural motion-blurred adversarial example to fool the deep neural network-based classification. Specifically, ABBA simulates the motion by adversarially shifting the object and background, neglecting the scene's natural motion. Unlike ABBA, our approach focuses on visual object tracking with real object movement indicated by two adjacent frames. Recently, some techniques [6; 25; 106] have been proposed to counter the interference of the environment. To this end, our method is proposed to better evaluate these VOTs' robustness.

2.4 Adversarial Attack

Extensive works have proved that state-of-the-art deep neural networks are still vulnerable to adversarial attacks by adding visually imperceptible noises or natural degradation to original images [38; 102; 15; 36; 49]. FGSM [38] perturbs normal examples along the gradient direction via the fast gradient sign method. MI-FGSM [30] integrates momentum terms into the iterative process that can help stabilize the update directions. C&W [9] introduces three new attacks for different norms (L_0, L_2, L_∞) through iterative optimization. However, the above methods cannot meet the real-time requirements due to the limited speed [51]. To realize the efficient attacking, [124; 119] propose one-step attacks by offline training on the targeted model. However, these methods are designed for classification and cannot attack trackers directly.

2.5 Backdoor Attack

We briefly review some famous poisoning-based backdoor attack works. *BadNets* [40] is the first backdoor attack technique that simply pastes a square-like trigger at the corner of the input image. Subsequently, more imperceptible and diverse trigger design patterns have been explored [89; 74; 90]. [89] proposes *WaNet*, which produces an invisible trigger by warping images with a particular direction. [74] introduces *image steganography* [101] to generate sample-specific invisible additive noises as backdoor triggers, and it can thus counter the existing backdoor defenses. [90] also uses a *U-net* architecture model to output the input-aware triggers. On the other hand, a few works [27; 98] attempt to manipulate the latent representation of the model to achieve more stealthy backdoor attacks. This inspires us to implement an NCE loss in Sect. 5.3.1 to improve the stealthiness of our backdoor attack.

2.6 Backdoor Defence

Backdoor defenses have been studied to detect or erase hidden backdoors that can be divided into two categories:

Poisoned data detection. Backdoor defenses have been studied to detect, prevent or erase hidden backdoors; these works can be divided into three categories:

- **Poisoned Data Detection.** In this line of work, the defense techniques use different strategies to detect whether the model is poisoned. For example, [10; 37; 18] rely on comparing the properties of the clean and the poisoned data. *SentiNet* [18] attempts to characterize the classification boundaries for poisoned and benign data. In particular, *STRIP* [37] takes a strategy that observes the randomness of the model inference under input perturbations: if the result representation is constant, the input image is poisoned; otherwise, it is not.
- **Secure Training.** A few works attempt to design secure training methods on poisoned training data, to obtain a model with high clean accuracy and a low attack success rate. For example, the ABL [75] utilized the observation that the training loss of poisoned data decreased much faster than that of clean data in training, such that the poisoned samples could be filtered and then unlearned to remove the backdoor from the model. The DBD method [60] found that the poisoned data will gather together in the feature space of the backdoored model, such that they could be classified to the same class, *i.e.*, injecting the backdoor into the model. To prevent the gathering of poisoned data, a three-stage training method was proposed, using self-supervised learning to learn the model backbone, then learning the classifier and filtering the poisoned data, followed by semi-supervised fine-tuning. The work [11] observed that the poisoned data is more sensitive to spatial transformations than the clean data in the feature space of the backdoored model, such that an effective sensitivity metric was designed to filter poisoned samples from the training dataset accurately. Their backdoor effects could be prevented in the training.
- **Backdoor Removal.** Another line of works [80; 125; 76; 116] uses clean data to reconstruct the model to remove possible backdoors. Distillation [76] is a framework that can effectively mitigate the effects of triggers. Pruning [94; 81; 71] initiates the idea of reducing the model size by removing specific channels of the neural network. The main idea of *Fine-Pruning* [80] is that the poisoned data will activate different channels compared to clean ones; due to that, pruning the neurons in the last layer of the backbone based on pure samples can remove the

backdoor attack effectively.

More evaluations and analyses of recent backdoor attacks and defenses could be found in the latest benchmark, called *BackdoorBench* [115]. In this work, we choose two typical techniques, namely STRIP and Fine-Pruning, as two reference defense techniques for designing the defense-aware version of the proposed framework TAT in Sect. 5.3.3.

We, therefore, choose STRIP and Fine-Pruning to check the robustness of TAT-BA.

2.7 Attacks to Visual Object Tracking

2.7.1 Adversarial Attack on Trackers

More recently, some works have been proposed to attack visual object tracking. PAT [114] generates physical adversarial textures via a white-box attack. SPARK [51] studies how to adapt existing adversarial attacks on tracking. OAA [13] proposes to add adversarial perturbations on the template at the initial frame. CSA [124] raises a one-step method and makes objects invisible to trackers by forcing the predicted bounding box to shrink. And EAA [78] try to train an adversarial network offline to attack the trackers.

Unlike the above works, we employ motion blur to perform adversarial attacks. Our work is designed to address three challenges: how to synthesize natural motion blur that meets the motion of the object and background in the video; how to make the blurred frame fool the state-of-the-art trackers easily; how to perform the adversarial blur attack efficiently.

Moreover, the above attacks need additional time and resources to generate adversarial examples. Furthermore, they conduct experiments in only digital settings. We will show our TAT can not only be implemented with several FPS declines compared with the original trackers (A.2.3) but also stronger destructive to trackers in Sect. 5.4.4 and achievable in real-world settings (Sect. 5.4.5).

2.7.2 Backdoor Attack on Trackers

FSBA [77] is the first work that achieves backdoor attacks on VOT. It tries to use a BadNet-pattern [40] trigger to drive the feature of poisoned data away from the original one and proved quite destructive. Although FSBA and our framework TAT target a similar problem setting, these techniques have several major differences. First, FSBA is designed to destroy the trackers in an untargeted attack manner, while our TAT controls the trackers with a specific target. Second, the trigger we designed is more invisible than the one in [77], and thus we could achieve better stealthiness. Third, TAT has a more destructive ability in Sect. 5.4.4.

3 | Online Auto Data Augmentation for Robust Visual Object Tracking

3.1 Introduction

With the advent of deep learning, visual object tracking has made remarkable progress, and the capabilities of trackers have improved significantly. However, the improvements achieved through model architecture enhancements have reached a bottleneck, and further progress in visual tracking requires a focus on other areas of research; for example, how to effectively use training samples from historical frames can bring new hope.

This section introduces an efficient online data augmentation method specifically designed for online updating object tracking models. In Sec. 3.2, we explain in detail from the initial idea to the preliminary validation to the final MixNet proposal. Also, some key training details are listed in Sec. 3.2.4, which is very important for Reproducibility. In Sec. 3.3, we evaluate our proposed DeepMix on three tracking benchmarks with three fashionable trackers. In the ablation study, our method with different backbones still has an enhancing effect.

3.2 Method

In this section, we first discuss the background and motivation of this work and formulate the online data augmentation for tracking in Sec. 3.2.1 and 3.2.2. Then, we propose the *MixNet* in Sec. 3.2.3 to realize effective and efficient online data augmentation for tracking. Finally, we detail how to embed MixNet into state-of-the-art trackers (*i.e.*, SiamRPN++ [69], DSiam [46; 50], and DiMP [5]) in Sec. 3.2.4.

3.2.1 Background and Motivation

Given a live video $\mathcal{V} = \{\mathbf{I}_t\}_{t=1}^{T}$ having T frames and the object bounding box annotated at the first frame (*i.e.*, $\mathbf{b}_1$), we aim to estimate the object's position and size at the subsequent $T-1$ frames. Most state-of-the-art methods complete this task by maintaining an object model for matching it with the subsequent frames. In general, we formulate the object localization at t-th frame as

$$\mathbf{p}_t = [\arg\min_{\mathbf{p}} \mathbf{M}_t[\mathbf{p}] = \arg\min_{\mathbf{p}} \mathrm{loc}(\varphi(\mathbf{I}_t), \theta_t) \tag{3.1}$$

where the $\mathbf{M}_t$ is a heat map whose the maximum (*i.e.*, $\mathbf{M}_t[\mathbf{p}_t]$) indicates the object's position in the frame $\mathbf{I}_t$, and it can be calculated by the $\varphi(\mathbf{I}_t)$ and θ_t where $\varphi(\cdot)$ is the backbone network for extracting embedding.

The object model θ_t determines the localization accuracy, which is initialized at the first frame and updated at subsequent frames. For example, in the popular Siamese network-based trackers [2; 69], the object model is constructed by using the embedding of the object at the first frame, *i.e.*, $\theta_t = \varphi(\mathbf{I}_1)$, and the localization is implemented by using cross-correlation, *i.e.*, $\mathrm{loc}(\varphi(\mathbf{I}_t), \theta_t) = \varphi(\mathbf{I}_t) * \varphi(\mathbf{I}_1)$. More recently, [5] proposes DiMP that uses an online updated classifier for localization (*i.e.*, the $\mathrm{loc}(\cdot)$ is set as a convolution layer).

The state-of-the-art trackers (*e.g.*, DiMP [5] and DSiam [46]) online update the object model to adapt object and background appearance variation. In general, we formulate the object model updating via

$$\theta_{t+1} = \mathrm{update}(\theta_t, \mathcal{X}_t) \tag{3.2}$$

Where $\mathcal{X}_t$ denotes the set of training samples that are cropped from the historical frames in which the objects are previously detected, for example, DSiam [46] proposes to online update the object model of the Siamese network via a transformation that is learned from the previous frame. DiMP [5] updates the classifier's parameters through a pre-trained model predictor that inputs 50 samples from previous frames.

Note that the updating process is a typical learning module, and recent works have demonstrated that data augmentation is important for enhancing image classification accuracy under various interferences [56]. Following similar ideas, we aim to explore how to online augment effective training samples, *i.e.*, $\mathcal{X}_t$, for the state-of-the-art trackers with existing updating methods, *i.e.*, online transformation in DSiam [46] and model predictor in DiMP [5].

3.2.2 Online Data Augmentation for Tracking

A simple way for online data augmentation is to borrow the existing techniques and conduct augmentation on the collected historical training samples, *i.e.*, $\mathcal{X}_t$, through

$$\hat{\mathcal{X}}_t = \mathrm{aug}(\mathcal{X}_t, \mathcal{T}) \tag{3.3}$$

where $\mathcal{T}$ denotes the set of sample-level transformations (*e.g.*, adding noise, blur, and rain) and $\hat{\mathcal{X}}_t$ is the augmented samples. The state-of-the-art data augmentation techniques are usually employed in the offline training process with random and diverse degradation-related sample-level transformations [56]. For example, AugMix [56] conducts multiple random augmentations for a raw sample and mixes them up for training the image classifiers. However, these methods could not adapt to the online data augmentation for visual object tracking directly due to the following reasons: ❶ the random sample-level transformations would generate new samples that require high time costs to extract their deep features, slowing down the trackers significantly. ❷ the transformations are based on degradation factors (*e.g.*, noise, blur, fog, rain, *etc.*) that can corrupt the raw samples, leading to the less discriminative object model.

To address the challenges mentioned above, we propose online data augmentation on embeddings of training samples. Specifically, we first extract embeddings of training samples in $\mathcal{X}_t$ and get $\varphi(\mathbf{I}_i) \in \mathbb{R}^{C\times W\times H}|\mathbf{I}_i \in \mathcal{X}_t\}$. Then, we concatenate all embeddings and obtain a tensor $\mathbf{X}_t \in \mathbb{R}^{N\times C\times W\times H}$ where N is the number of training samples in $\mathcal{X}_t$. Our goal is to map the tensor $\mathbf{X}_t$ to a new counterpart denoted as $\hat{\mathbf{X}}_t \in \mathbb{R}^{K\times C\times W\times H}$ that can be fed into existing updating modules to produce a more effective object model. Note that performing augmentation on the embedding level is much more efficient than on the sample level, alleviating the first challenge. Regarding the second challenge, we mix embeddings of all samples with the guidance of previous localization results. Intuitively, the interested object might be at any position in the scene during the video-capturing process, and it is reasonable to augment the training samples by putting the object in possible background regions. To this end, given $\mathbf{X}_t$ and the detected bounding boxes $\{\mathbf{b}_i \in \mathbb{R}^{4\times 1}|i = 1, \ldots, N\}$ of N training samples, we can split the samples to object and background regions and mixing them up to produce new samples. We formulate this process by

$$\hat{\mathbf{X}}_t = (\mathbf{W}_{\mathrm{o}} \circledast \mathbf{X}_t) \odot \mathbf{M}_{\mathrm{o}} + (\mathbf{W}_{\mathrm{b}} \circledast \mathbf{X}_t) \odot (1 - \mathbf{M}_{\mathrm{o}}), \tag{3.4}$$

where $\mathbf{M}_{\mathrm{o}} \in \mathbb{R}^{N\times C\times W\times H}$ are binary masks for the N training samples where the elements within the object regions are set to one while others are zero. The object regions are obtained according to the detection results $\{\mathbf{b}_i \in \mathbb{R}^{4\times1} | i = 1, ..., N\}$ and the '$\odot$' denotes the element-wise multiplication. Besides, the '$\circledast$' denotes the convolution layer while the tensor $\mathbf{X}_t$ is filtered by $\mathbf{W}_{\{\text{o or b}\}} \in \mathbb{R}^{K\times N\times3\times3}$ with the padding hyper-parameter to be one. The variable K denotes the number of samples in the output tensor. More specifically, for the cth channel of $\mathbf{X}_t$ (*i.e.*, $\mathbf{X}_t[c] \in \mathbb{R}^{N\times W\times H}$), we filter it with $\mathbf{W}_{\{\text{o or b}\}}$ and get the cth channel of $\hat{\mathbf{X}}_t[c] \in \mathbb{R}^{K\times W\times H}$. Intuitively, $\mathbf{W}_{\{\text{o or b}\}}$ indicates how to fuse the N training samples and get K new samples. The $\mathbf{W}_\text{o}$ and $\mathbf{W}_\text{b}$ take charge of mixing object and background regions, respectively.

However, to make the above method work, we should consider the following issues: ❶ how to estimate $\mathbf{W}_\text{o}$ and $\mathbf{W}_\text{b}$ to fit different cases? ❷ How to make the above module efficient? To alleviate these issues, we propose the *MixNet* that can produce augmented data in one step.

3.2.3 MixNet for Efficient Online Data Augmentation

We propose *MixNet* that takes the $\mathbf{X}_t$ as the input and predict the kernels $\mathbf{W}_\text{o}$ and $\mathbf{W}_\text{b}$ that are suitable for $\mathbf{X}_t$. We can use the pre-trained *MixNet* to generate the kernels in a one-step way, leading to efficient online data augmentation. We show the architecture of *MixNet* in Fig. 3.1. Specifically, MixNet contains two sub-networks for predicting the $\mathbf{W}_\text{o}$ and $\mathbf{W}_\text{b}$, respectively. The two sub-networks share the same architecture but have independent parameters. The architecture has three convolution layers with a kernel size of 3×3 and an averaging pooling layer. We can embed MixNet into diverse tracking frameworks by adequately setting their input and output channels.

3.2.4 Implementation for SOTA Trackers

In this part, we detail the way of using our method for three state-of-the-art trackers, *i.e.*, SiamRPN++ [70; 68], DSiam [46; 50], and DiMP [5]. Simply, we can embed *DeepMix* into these trackers by transforming their training samples and get $\hat{\mathbf{X}}_t$. Then, we mix it with the raw training samples

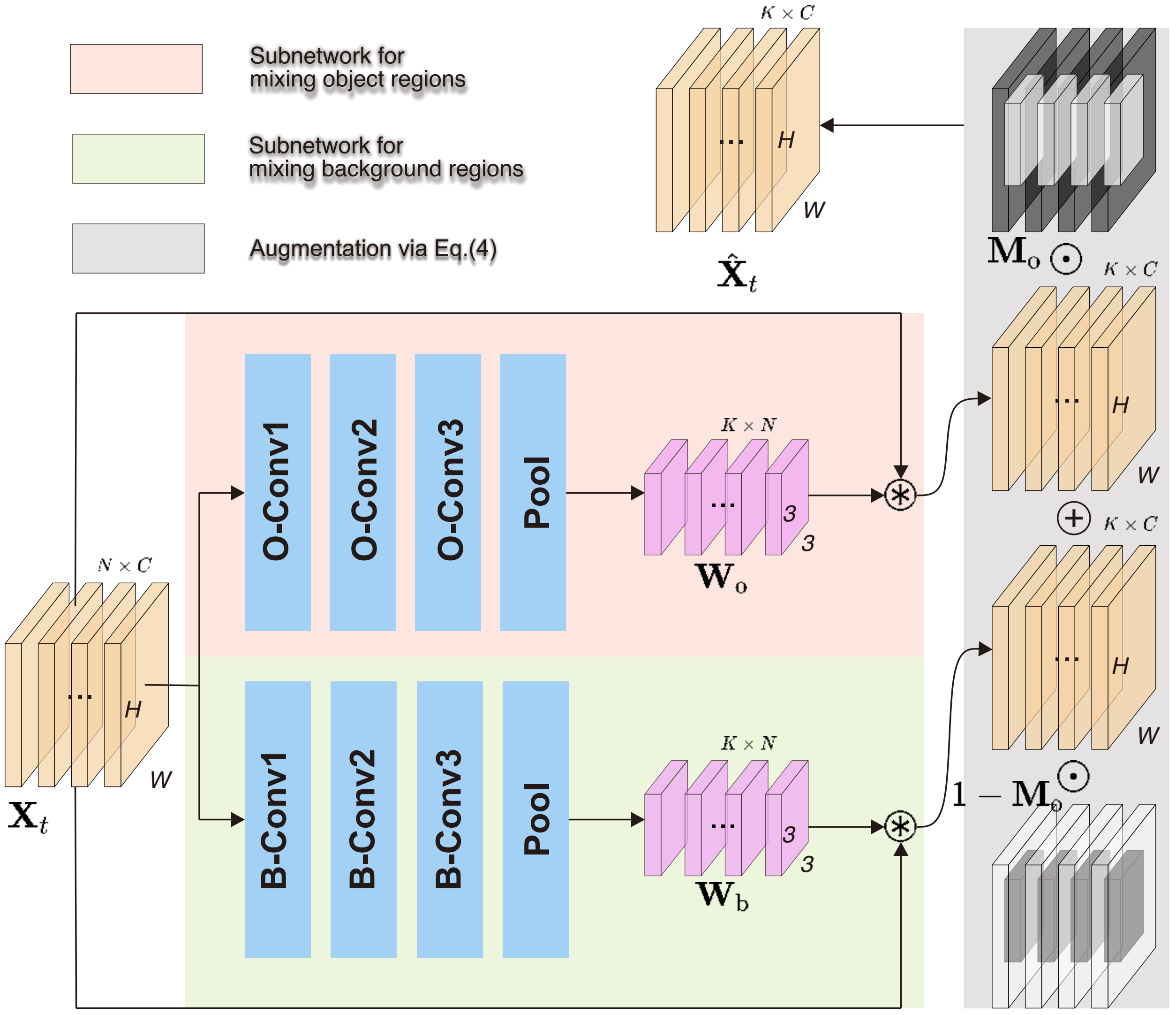

Figure 3.1: Architecture of MixNet that contains two sub-networks to estimate filtering parameters for mixing object and background regions, respectively.

(*i.e.*, $\mathbf{X}_t$) by $\alpha_1 \hat{\mathbf{X}}_t + \alpha_2 \mathbf{X}_t$) and feed it into the updating modules. For all examples, we fix $\alpha_1 = 0.05$ and $\alpha_1 = 0.8$ and discuss its influence in Sec. 3.3.3.

DSiam and SiamRPN++ with DeepMix. We collect historical samples ($\mathbf{X}_t$ size is 15x256x29x29) to generate the kernel ($\mathbf{W}_{\{\mathrm{o\ or\ b}\}}$ size is 15x1x3x3) and then filter with the features (its size is 1x256x29x29) of the current frame. Finally, DeepMix output the new samples ($\hat{\mathbf{X}}_t$ size is 1x256x29x29).

Table 3.1: Comparison with SOTA Trackers under OPE setup.

Dataset	OTB-2015		LaSOT	
Metrics	AUC	Prec	Success	Prec
MLT [17]	0.611	-	0.368	-
GradNet [73]	0.639	0.861	0.365	0.351
ATOM [23]	0.667	0.879	0.514	0.505
SiamDW [17]	0.674	-	0.384	-
POST [108]	0.678	0.907	0.481	0.463
CRPN [33]	0.675	-	0.455	-
ASRCF [21]	0.692	0.922	0.359	0.337
MAML [107]	0.712	-	0.523	-
DSiam	0.646	0.845	0.438	0.431
DSiam-DeepMix	**0.658**	**0.861**	**0.439**	**0.431**
SiamRPN++	0.650	0.853	0.447	0.446
SiamRPN++-DeepMix	**0.663**	**0.870**	**0.459**	**0.463**
DiMP	0.660	0.859	0.532	0.532
DiMP-DeepMix	**0.683**	**0.890**	**0.536**	**0.538**

DiMP with DeepMix. DiMP stores samples to train the classifier; we directly input these samples ($\mathbf{X}_t$ size is 50x256x22x22) and obtain new samples with the same size as the Fig. 3.1 illustrate.

We directly train MixNet along with its embedded trackers. We can simply use the targeted trackers' original training program and training data to train their own MixNets. We make some minor modifications to MixNet to adapt to different trackers.

Training details. For DSiam and SiamRPN++, the original training program uses a pair of images (*i.e.*, template and search region) as a training sample. For each sample, We apply data augmentation strategies on a template to construct a training set containing 15 samples. We input them into MixNet and generate a new sample to mix with the original template. We implement the SGD optimizer with a weight decay of 0.0005, base lr of 0.005, and momentum of 0.9. We train the MixNet for 40 epochs and 6000 samples per epoch.

Regarding DiMP, its training program picks up three images from each video as training samples for its model predictor. To match our MixNet during testing, we change it to 50 images from each video for MixNet. We apply SGD optimizer with weight decay of 0.0005 for all parameter layers, base lr of 0.005, and momentum of 0.9.We train the MixNet for 50 epochs and 1000 videos per epoch.

3.3 Experiments

3.3.1 Setups

Datasets, metrics, and baseline. We evaluate our proposed DeepMix on three tracking benchmarks: VOT-2018 [65] (60 videos, 356 frames average length), LaSOT [32] (280 videos, 2448 frames average length), OTB-2015 [118] (100 videos, 590 frames average length). VOT-2018 uses a reset-based evaluation in which once the object is lost, the tracker is restarted with the ground truth box five frames later and gets a penalty. The main evaluation criterion is the expected average overlap (EAO) [66]. In general, the larger EAO indicates better performance. OTB-2015 and LaSOT only give the tracker the ground truth of the initial frame and obtain bounding box sequence, terms *one-pass evaluation* (OPE). The results are reported in Table 3.1. AUC represents the area under the curve of the success plot.

We compare our DeepMix with six top trackers on VOT2018, including DaSiamRPN [137], SiamMask [110], MAML [107], UpdateNet [131], ATOM [23],SiamDW [133]. We also choose some excellent trackers on OTB2015, such as MLT [17], GradNet [73],POST [108], CRPN [33], ASRCF [21].[16]

3.3.2 State-of-the-art Comparison

We compare our proposed DeepMix with state-of-the-art methods on three challenging tracking benchmarks. Specifically, we employ ResNet18 [54] as the backbone for DiMP, AlexNet [67] as the backbone for DSiam, and MobileV2 [97] as the backbone for SiamRPN++ on account of DeepMix's extreme improvement for a simple network.

Table 3.2: Comparison with SOTA Trackers on VOT2018

Metrics	EAO	Accuracy	Robustness
DaSiamRPN [137]	0.383	0.586	0.276
SiamMask [110]	0.387	0.642	0.295
MAML [107]	0.392	0.635	0.220
UpdateNet [131]	0.393	-	-
ATOM [23]	0.401	0.590	0.204
SiamDW [133]	0.405	0.597	0.234
DSiam	0.266	0.577	0.421
DSiam-DeepMix	**0.287**	**0.58**	**0.407**
SiamRPN++	0.348	0.583	0.29
SiamRPN++-DeepMix	**0.405**	**0.597**	**0.234**
DiMP	0.214	0.578	0.553
DiMP-DeepMix	**0.234**	**0.612**	**0.51**

Table 3.3: Ablation analysis of different backbones

Metrics	AUC	Precision	NormPrecision
DiMP (ResNet18)	0.660	0.859	0.807
DiMP (ResNet18)-DeepMix	**0.683**	**0.890**	**0.834**
DiMP (ResNet50)	0.684	0.894	0.842
DiMP (ResNet50)-DeepMix	**0.694**	**0.900**	**0.852**

OTB-2015 and LaSOT. We report results on the OTB-2015 and LaSOT datasets in Table 3.1. Results show that: ❶ DeepMix improves all of its original trackers. ❷ DeepMix has a significant improvement of 2.3 percentage points with AUC and raises the ranking of DiMP from fifth to third compared with other trackers on OTB-2015. DiMP-DeepMix also achieves the top success score on LaSOT. ❸ DiMP collects historical samples' embeddings and trains the predictor online. Our MixNet is designed for data augmentation; thus DeepMix has better compatibility with DiMP and achieves more improvement than DSiam and SiamRPN++.

VOT2018. We report results on the VOT2018 on Table 3.2. DeepMix with three trackers still be in effect. DiMP employs 250 samples for online training on VOT2018(Not 50 on other datasets) that cost extreme memory with DeepMix; we still implement the same hyperparameters as on OTB-2015. Although it has achieved much lower results than reported, it also proves the effectiveness of DeepMix. SiamRPN-DeepMix obtain a striking 0.057 improvement on EAO. Even though it uses MobileV2 as its backbone, it achieves state-of-the-art on VOT2018.

3.3.3 Ablation study

DeepMix with different backbones. We present the result on OTB-2015 dataset in Table 3.3 with different backbones. Number 18 and 50 means ResNet50 and ResNet18 [54], separately. The

Table 3.4: Comparing three variants of our method, i.e., DeepMix-Opt, DeepMix-single, and DeepMix, on DiMP tracker and OTB-2015 dataset to validate the effectiveness of MixNet.

Metrics	AUC	Prec.	NormPrec.	FPS
DiMP	0.660	0.859	0.807	27.0
DiMP-DeepMix-Opt	**0.667**	**0.873**	**0.816**	**10**
DiMP-DeepMix-single	**0.676**	**0.884**	**0.831**	**26.5**
DiMP-DeepMix	**0.683**	**0.890**	**0.834**	**26.0**

w and w/o mean model with or without DeepMix. It shows that: ❶ DeepMix can take a stable effect for any network architecture. ❷ DeepMix have more improvement on ResNet18-based than ResNet50-based model. It is probably because more powerful networks are less dependent on DeepMix.

Validation of MixNet. We implement a naive data augmentation method as the baseline to validate the effectiveness of MixNet. That is, we calculate the filtering parameters, *i.e.*, $\mathbf{W}_\mathrm{o}$ and $\mathbf{W}_\mathrm{b}$, by online optimizing an objective function via the gradient descent to replace the proposed MixNet. Specifically, we define an objective function: the L_2 distance between a predicted heat map and a Gaussian map with the highest score on the detected position. Then, at tth frame, we can minimize the objective function by tuning the $\mathbf{W}_\mathrm{o}$ and $\mathbf{W}_\mathrm{b}$ via the gradient descent for ten iterations. We denote this method as 'DeepMix-Opt' and compare it with the final version DeepMix based on the DiMP tracker and OTB-2015 dataset. As shown in Table 3.4, DeepMix-Opt via online iterative optimization can also enhance the tracking accuracy but immensely increase the computational cost, slowing down the DiMP significantly.

As Fig. 3.1 show, MixNet has two branches and output two filters ($\mathbf{W}_\mathrm{o}$ and $\mathbf{W}_\mathrm{b}$). We test another version of MixNet: keep only one branch and output one convolution kernel, then filters with samples $\mathcal{X}_t$, regardless of object or background. We term it as DeepMix-single. As shown in Table 3.4, DeepMix-single outperforms DiMP with a competitive speed, but it still is weaker than the final version DeepMix. Therefore, learning different patterns of objects or backgrounds is an essential method for online data augmentation. In contrast, DeepMix with the MixNet achieves much higher accuracy improvement with only one FPS speed decrease, demonstrating the effectiveness and advantages of the proposed MixNet.

3.3.4 Conclusion

In this chapter, we have taken a deep dive into the data augmentation aspect for improving online visual object tracking, a long-overlooked facet in this domain. Specifically, we have proposed the DeepMix as a complete pipeline that takes historical samples' embeddings as input and generates augmented online, thus enhancing the state-of-the-art online learning methods for visual object tracking. To this end, we have proposed the *online data augmentation* for tracking that online augments the historical samples through object-aware filtering. Then, we have further proposed the *MixNet*, an offline trained deep neural network for performing online data augmentation within one step, boosting the tracking accuracy while preserving high speeds of the state-of-the-art online learning methods. We have conducted extensive experiments on three different tracking frameworks, *i.e.*, DiMP, DSiam, and SiamRPN++, and on three large-scale and challenging datasets, *i.e.*, OTB-2015, LaSOT, and VOT. The experimental results have demonstrated and verified the effectiveness and advantages of the proposed method.

4 | Learning to Adversarially Blur Visual Object Tracking

4.1 Introduction

The method of the previous chapter is primarily suitable for online-updated trackers, which restricts its applicability to a certain extent. To further enhance the robustness of the tracking model, we consider the improvement from its testing aspect.

Most existing benchmarks merely indicate the presence or absence of motion blur in videos or frames. However, this limited information is inadequate for comprehensive analysis, as it fails to control other variables or eliminate interference from different degradation modes. Consequently, drawing conclusive findings about the effects of motion blur from these benchmarks can be incomplete. Furthermore, although current datasets are extensive, they do not sufficiently encompass the diverse range of motion blur encountered in real-world scenarios. Motion blur is a dynamic and unpredictable phenomenon resulting from both camera and object movements within a scene. Existing motion blur generation methods cannot fully capture malicious or unintentional threats in visual object tracking. These methods can only produce natural motion blur, which does not effectively expose the vulnerabilities of visual object trackers.

It is necessary to explore a novel motion blur synthetic method for analyzing the robustness of the visual object trackers. In this section, we explore a new motion blur evaluation criterion. In Sec. 4.2, we first introduced the synthetic motion blur approach for visual object tracking. Then, optimization-based and one-step methods are utilized to implement adversarial attacks in motion blur. In Sec. 4.3, we conduct the experiment on four popular datasets to verify the offensive nature of Adversarial Blur Attacks. We also compare transferability with other adversarial attacks.

4.2 Adversarial Blur Attack against Tracking

In this section, we first study how to synthesize natural motion blur under the visual tracking task in Sec. 4.2.1 and summarize the variables that should be solved to perform attacks. Then, we propose the optimization-based ABA (OP-ABA) in Sec. 4.2.2 with a novel objective function to guide the motion blur generation via the iterative optimization process. To allow high-efficiency attacks for real-time trackers, we further propose one-step ABA (OS-ABA) in Sec. 4.2.3 by training a newly designed *joint motion and kernel predictive network* under the supervision of the objective function of OP-ABA. Finally, we summarize the attacking details with OP-ABA and OS-ABA in Sec. 4.2.4.

4.2.1 Motion Blur Synthesizing for Visual Tracking

In a typical tracking process, given the t-th frame $\mathbf{I}_t$ of a live video and an object template specified at the first frame, a tracker uses a pre-trained deep model $\phi_{\theta_t}(\mathbf{I}_t)$ to predict the location and size of the object (*i.e.*, the bounding box tightly wrapping the thing) in this frame where θ_t denotes the template-related parameter and can be updated during the tracking process. For the adversarial blur attack, we aim to generate a motion-blurred counterpart of $\mathbf{I}_t$, which can fool the tracker into estimating the incorrect bounding box of the object while having the natural motion-blur pattern.

To this end, we review the generation principle of realistic motion blur [87; 91; 8; 49; 45]: the camera sensor captures an image by receiving and accumulating light during the shutter procedure. The light at each time can be represented as an instant image, and there are a series of instant images for the shutter process. When the objects or background move, the light accumulation will cause blurry effects, which can be approximated by averaging the instant images.

Under the above principle, when we want to blur $\mathbf{I}adversarially_t$, we need to do two things: *First*, synthesizing the instant images during the shutter process and letting them follow the motion of the object and background in the video; *Second*, accumulating all instant images to get the motion-blurred $\mathbf{I}_t$. The main challenge is making the two steps adversarially tunable to fool the tracker easily while preserving the natural motion blur pattern.

For the first step, we propose to generate the instant images under the guidance of the optical flow $\mathbf{U}_t$ that describes the pixel-wise moving distance and direction between $\mathbf{I}_t$ and its neighbor $\mathbf{I}_{t-1}$. Specifically, given two neighboring frames in a video, *e.g.*, $\mathbf{I}_{t-1}$ and $\mathbf{I}_t$, we regard them as the start and end time stamps for camera shutter process, respectively. Assuming there are N instant images, we denote them as $\{\mathbf{I}_t^i\}_{i=1}^N$ where $\mathbf{I}_t^1 = \mathbf{I}_{t-1}$ and $\mathbf{I}_t^N = \mathbf{I}_t$. Then, we calculate the optical flow $\mathbf{U}_t$ between $\mathbf{I}_{t-1}$ and $\mathbf{I}_t$ and split it into $N-1$ sub-motions, *i.e.*, $\{\mathbf{U}_t^i\}_{i=1}^{N-1}$ where $\mathbf{U}_t^i$ represents the optical flow between $\mathbf{I}_t^i$ and $\mathbf{I}_t^{i+1}$. We define $\mathbf{U}_t^i$ as a scaled $\mathbf{U}_t$ with pixel-wise ratios (*i.e.*, $\mathbf{W}_t^i$)

$$\mathbf{U}_t^i = \mathbf{W}_t^i \odot \mathbf{U}_t, \tag{4.1}$$

where $\mathbf{W}_t^i$ has the same size with $\mathbf{U}_t$ and $\odot$ denotes the pixel-wise multiplication. All elements in $\mathbf{W}_t^i$ range from zero to one and we constraint the summation of $\{\mathbf{W}_t^i\}_{i=1}^N$ at the same position to be one, *i.e.*, $\forall \mathbf{p}, \sum_i^{N-1} \mathbf{W}_t^i[\mathbf{p}] = 1$ where $\mathbf{W}_t^i[\mathbf{p}]$ denotes the $\mathbf{p}$-th element in $\mathbf{W}_t^i$. Note that, the ratio matrices, *i.e.*, $\{\mathbf{W}_t^i\}_{i=1}^N$, determine the motion pattern. For example, if we have $\forall \mathbf{p}$, $\{\mathbf{W}_t^i[\mathbf{p}] = \frac{1}{N-1}\}_{i=1}^{N-1}$ and can calculate the sub-motions by $\{\mathbf{U}_t^i = \frac{1}{N-1}\mathbf{U}_t\}_{i=1}^{N-1}$, all pixels follow the uniform motion.

With Eq. (4.1), we get all sub-motions (*i.e.*, $\{\mathbf{U}_t^i\}_{i=1}^{N-1}$) and produce the instant images by warping $\mathbf{I}_t$ w.r.t. different optical flows. For example, we synthesize $\mathbf{I}_t^i$ by

$$\mathbf{I}_t^i = \frac{1}{2}\text{warp}(\mathbf{I}_{t-1}, \sum_{j=1}^{i-1} \mathbf{W}_t^j \odot \mathbf{U}_t^j) + \frac{1}{2}\text{warp}(\mathbf{I}_t, \sum_{j=i}^{N-1} \mathbf{W}_t^j \odot \mathbf{U}_t^j), \tag{4.2}$$

where $\sum_{j=1}^{i-1} \mathbf{W}_t^j \odot \mathbf{U}_t^j$ represents the optical flow between $\mathbf{I}_{t-1}$ and $\mathbf{I}_t^i$ while $\sum_{j=i}^{N-1} \mathbf{W}_t^j \odot \mathbf{U}_t^j$ denotes the optical flow between $\mathbf{I}_t^i$ and $\mathbf{I}_t$. The function warp$(\cdot)$ is to warp the $\mathbf{I}_{t-1}$ or $\mathbf{I}_t$ according to the corresponding optical flow and uses the implementation in [49] for spatial transformer network.

For the second step, after getting $\{\mathbf{I}_t^i\}_{i=1}^N$, we can synthesize the motion-blurred $\mathbf{I}_t$ by summing up the N instant images with pixel-wise accumulation weights $\{\mathbf{A}_i\}_{i=1}^N$

$$\hat{\mathbf{I}}_t = \sum_{i=1}^{N} \mathbf{A}_t^i \odot \mathbf{I}_t^i. \tag{4.3}$$

where $\mathbf{A}_t^i$ has the same size with $\mathbf{I}_t^i$ and all elements range from zero to one. For simulating realistic motion blur, all elements of $\mathbf{A}_t^i$ are usually fixed as $\frac{1}{N}$, which denotes the accumulation of all instant images.

Overall, we represent the whole blurring process via Eqs. (4.3) and (4.2) as $\hat{\mathbf{I}}_t = \text{Blur}(\mathbf{I}_t, \mathbf{I}_{t-1}, \mathcal{W}_t, \mathcal{A}_t)$. To perform adversarial blur attack for the frame $\mathbf{I}_t$, we need to solve two sets of variables, *i.e.*, $\mathcal{W}_t = \{\mathbf{W}_t^i\}_{i=1}^{N-1}$ determining the motion pattern and $\mathcal{A}_t = \{\mathbf{A}_t^i\}_{i=1}^N$ deciding the accumulation strategy. In Sec. 4.2.2, we follow the existing adversarial attack pipeline and propose the optimization-based ABA by defining and optimizing a tracking-related objective function to get $\mathcal{W}_t$ and $\mathcal{A}_t$. In Sec. 4.2.3, we design a network to predict $\mathcal{W}_t$ and $\mathcal{A}_t$ in a one-step way.

4.2.2 Optimization-based Adversarial Blur Attack

This section proposes solving $\mathcal{W}_t$ and $\mathcal{A}_t$ by optimizing the tracking-related objective function. Specifically, given the original frame $\mathbf{I}_t$, a tracker can estimate a response or classification map by $\mathbf{Y}_t = \phi_{\theta_t}(\mathbf{I}_t)$ whose maximum indicates the object's position in the $\mathbf{I}_t$. Our attack aims to generate a blurred $\mathbf{I}_t$ (*i.e.*, $\hat{\mathbf{I}}_t = \text{Blur}(\mathbf{I}_t, \mathbf{I}_{t-1}, \mathcal{W}_t, \mathcal{A}_t)$) to let the predicted object position indicated by $\hat{\mathbf{Y}}_t = \phi_{\theta_t}(\hat{\mathbf{I}}_t)$ be far away from the original one indicated by $\mathbf{Y}_t$.

To this end, we optimize $\mathcal{W}_t$ and $\mathcal{A}_t$ by minimizing

$$\begin{aligned} &\underset{\mathcal{W}_t, \mathcal{A}_t}{\arg\min} J(\phi_{\theta_t}(\text{Blur}(\mathbf{I}_t, \mathbf{I}_{t-1}, \mathcal{W}_t, \mathcal{A}_t)), \mathbf{Y}_t^*) \\ &\text{subject to } \ \forall \mathbf{p}, \forall i, \sum_i^{N-1} \mathbf{W}_t^i[\mathbf{p}] = 1, \sum_i^{N} \mathbf{A}_t^i[\mathbf{p}] = 1, \end{aligned} \tag{4.4}$$

where the two constraints on $\mathbf{W}_t^i$ and $\mathbf{A}_t^i$ make sure the synthetic motion blur does not have obvious distortions. The function $J(\cdot)$ is a distance function set as L_2. The regression target $\mathbf{Y}_t^*$ denotes the desired response map and is obtained under the guidance of the original $\mathbf{Y}_t$. Specifically, with the original response map $\mathbf{Y}_t$, we know the object's position and split $\mathbf{Y}_t$ into two regions, the object region, and background region, according to the object size. Then, we can find the position (*e.g.*, $\mathbf{q}$) having the highest response score at the background region of $\mathbf{Y}_t$, and then we set $\mathbf{Y}^*[\mathbf{q}] = 1$ and other elements of $\mathbf{Y}^*$ to be zero. Note that the above setup is suitable for regression-based trackers, *e.g.*, DiMP and KYS, and can be further adapted to attack classification-based trackers, e.g., SiamRPN++, by setting $J(\cdot)$ as the cross-entropy loss function and $\mathbf{Y}^*[\mathbf{q}] = 1$ with its other elements to be -1.

Following the common adversarial attacks [39; 31; 51; 49], we can solve Eq. (4.4) via the signed gradient descent and update the $\mathcal{W}_t$ and $\mathcal{A}_t$ iteratively with specified step size and iterative number. We show the synthesized motion blur of OP-ABA in Fig. 4.1. OP-ABA can synthesize natural

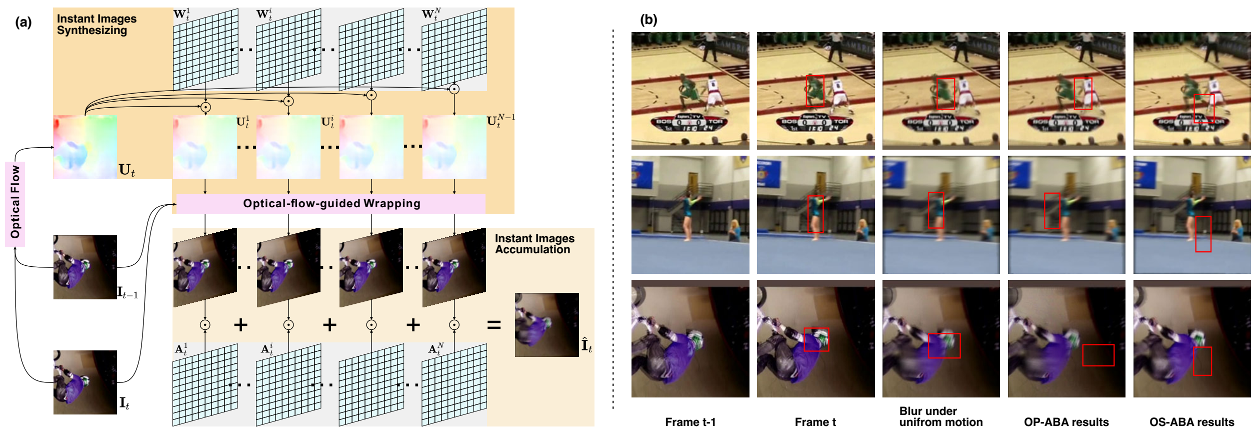
(a)
Instant Images Synthesizing
Optical Flow
Optical-flow-guided Wrapping
Instant Images Accumulation
(b)
Frame t-1
Frame t
Blur under unifrom motion
OP-ABA results
OS-ABA results

Figure 4.1: (a) shows the motion blur synthesizing process with two frames, *i.e.*, $\mathbf{I}_t$ and $\mathbf{I}_{t-1}$, and two sets of variables, *i.e.*, $\{\mathcal{A}_t^i\}$ and $\{\mathcal{W}_t^i\}$, should be determined for attacking. (b) shows three cases of normal blur under uniform motion, the OP-ABA blurring results, and OS-ABA blurring results.

motion-blurred frames that have a similar appearance to the normal motion blur.

4.2.3 One-Step Adversarial Blur Attack

To allow efficient adversarial blur attack, we propose to predict the motion and accumulation weights (*i.e.*, $\mathcal{W}_t$ and $\mathcal{A}_t$) with a newly designed network denoted as *joint adversarial motion and accumulation predictive network (JAMANet)* in a one-step way, which is pre-trained through the objective function Eq. (4.4) and a naturalness-aware loss function. Specifically, we use JAMANet to process the neighboring frames (*i.e.*, $\mathbf{I}_t$ and $\mathbf{I}_{t-1}$) and predict the $\mathcal{W}_t$ and $\mathcal{A}_t$, respectively. Meanwhile, we also employ a pre-trained network to estimate the optical flow $\mathbf{U}_t$ between $\mathbf{I}_t$ and $\mathbf{I}_{t-1}$. Here, we use the PWCNet [100] since it achieves good results on diverse scenes. Then, with Eq. (4.2)-(4.3), we can obtain the motion-blurred frame $\hat{\mathbf{I}}_t$. After that, we feed $\hat{\mathbf{I}}_t$ into the loss functions and calculate gradients of parameters of JAMANet to perform optimization. We show the framework in Fig. 4.2.

Architecture of JAMANet. We first build two parameter sets with constant values, which are denoted as $\mathcal{A}_{\text{norm}} = \{\mathbf{A}^i_{\text{norm}}\}$ and $\mathcal{W}_{\text{norm}} = \{\mathbf{W}^i_{\text{norm}}\}$. All elements in $\mathcal{A}_{\text{norm}}$ and $\mathcal{W}_{\text{norm}}$ are

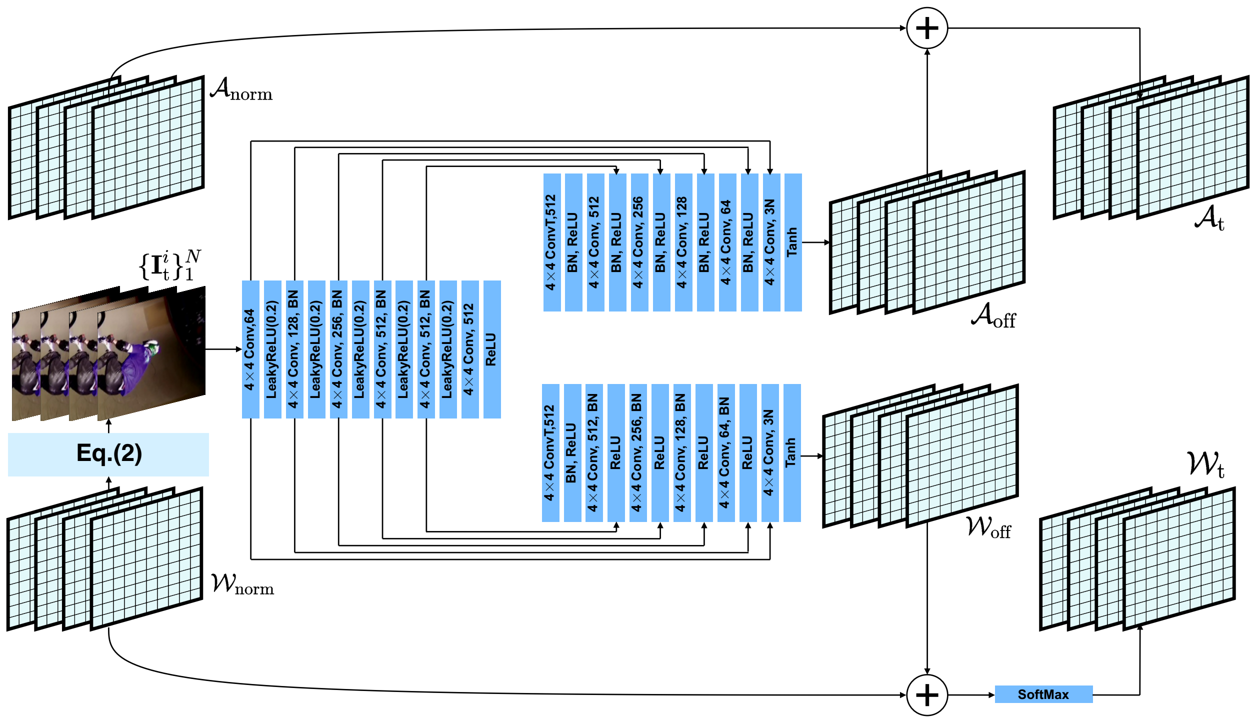

Figure 4.2: Architecture of JAMANet.

fixed as $\frac{1}{N}$ and $\frac{1}{N-1}$, respectively. We then use $\mathcal{W}_{\text{norm}}$, $\mathbf{I}_{t-1}$, and $\mathbf{I}_t$ to generate N instant images through Eq. (4.2). JAMANet is built based on the U-Net architecture [96] but contains two decoder branches, which are fed with the N instant images $\{\mathbf{I}_t^i\}_{i=1}^N$ and outputs the offsets w.r.t. $\mathcal{W}_{\text{norm}}$ and $\mathcal{A}_{\text{norm}}$. We name them as $\mathcal{W}_{\text{off}}$ and $\mathcal{A}_{\text{off}}$. The input $\{\mathbf{I}_t^i\}_{i=1}^N$ is size of $(N, 3, H, W)$. We resize it to $(1, 3N, H, W)$ and normalize the values from -1 to 1. The architecture is a fully convolutional encoder/decoder model with skip connections. In the encoder stage, we use six convolutions with the kernel size 4x4 and the LeakyReLU [82] activation function. Unlike the standard U-Net, JAMANet has two decoders. Specifically, one branch is set to estimate the $\mathcal{A}_{\text{off}}$, containing six transposed convolutions [128] with the latest activation function as Tanh. We can calculate the final $\mathcal{A}_t$ through $\mathcal{A}_t = \mathcal{A}_{\text{norm}} + \mathcal{A}_{\text{off}}$. Another branch is to predict $\mathcal{W}_{\text{off}}$ and get $\mathcal{W}_t = \mathcal{W}_{\text{norm}} + \mathcal{W}_{\text{off}}$. This architecture is the same as the previous one but follows a Softmax for catering to constraints of Eq. (4.4)[1].

Loss functions. We train the JAMANet with two loss functions:

$$\mathcal{L} = \mathcal{L}_{\text{adv}} + \lambda \mathcal{L}_{\text{natural}}, \tag{4.5}$$

Where the first loss function, *i.e.*, $\mathcal{L}_{\text{adv}}$, is set to the objective function in Eq. (4.4) to make sure the background content instead of the object is highlighted. Note that this loss function enhances the capability of adversarial attack, misleading the raw trackers. It, however, neglects the naturalness of adversarial blur. To this end, we set the loss function $\mathcal{L}_{\text{natural}}$ as

$$\mathcal{L}_{\text{natural}} = \sum_{i}^{N} \|\mathbf{A}_t^i - \mathbf{A}_{\text{norm}}^i\|_2. \tag{4.6}$$

This loss function encourages the estimated accumulation parameters to be similar to the normal ones, leading to natural motion blur.

Training details. We use GOT-10K [61] as our training dataset, which includes 10,000+ sequences and 500+ object classes. For each video in GOT-10K [61], we set the first frame as a template and take two adjacent frames as an image pair, *i.e.*, $(\mathbf{I}_{t-1}, \mathbf{I}_t)$. We select eight image pairs from each video. The template and two adjacent frames make up a training sample. Here, we implement the OS-ABA for attacking two trackers, *i.e.*, SiamRPN++ [69] with ResNet50 and MobileNetv2, respectively. The experiment shows that OS-ABA has strong transferability against other state-of-the-art trackers. During the training iteration, we first calculate the template's embedding to construct tracking model ϕ_{θ_t} and the original response map $\mathbf{Y}_t$ (*i.e.*, the positive activation map of SiamRPN++). Then, we get $\mathbf{Y}_t^*$ and initialize the blurred frame via $\text{Blur}(\mathbf{I}_t, \hat{\mathbf{I}}_{t-1})$. We can calculate the loss via Eq. (4.5) and obtain the gradients of the JAMANet via backprop-

[1]To let $\mathcal{A}_t$ also meet the constraints, for any pixel $\mathbf{p}$, we first select the element $j = \arg\min_{i, i\in[1,N]} \mathbf{A}_{\text{off}}^i[\mathbf{p}]$ and then set $\mathbf{A}_t^j[\mathbf{p}] = 1 - \sum_{i,i\neq j}^{N} \mathbf{A}_t^i[\mathbf{p}]$.

agation for parameter updating. We train the JAMANet for ten epochs, requiring 9 hours on a single Nvidia RTX 2080Ti GPU. We use the Adam [64] with the learning rate 0.0002 to optimize network parameters, and the loss weight λ equals 0.001.

Table 4.1: Attacking results of OP-ABA and OS-ABA against SiamRPN++ with ResNet50 and MobileNetv2 on OTB100 and VOT2018. The best results are highlighted by **red** color.

SiamRPN++	Attacks	OTB100				VOT2018	
		Org. Prec.	Prec. Drop ↑	Org. Succ.	Succ. Drop ↑	Org. EAO	EAO Drop ↑
ResNet50	OP-ABA	87.8	**41.7**	66.5	**31.2**	0.415	**0.375**
	OS-ABA	87.8	32.5	66.5	28.1	0.415	0.350
MobNetv2	OP-ABA	86.4	**49.6**	65.8	**37.6**	0.410	**0.384**
	OS-ABA	86.4	37.3	65.8	30.1	0.410	0.338

Table 4.2: Attacking results of OP-ABA and OS-ABA against SiamRPN++ with ResNet50 and MobileNetv2 on UAV123 and LaSOT. The best results are highlighted by **red** color.

SiamRPN++	Attacks	UAV123				LaSOT			
		Org. Prec.	Prec. Drop. ↑	Org. Succ.	Succ. Drop ↑	Org. Prec.	Prec. Drop ↑	Org. Succ.	Succ. Drop ↑
ResNet50	OP-ABA	80.4	**30.4**	61.1	**23.1**	49.0	**28.7**	49.7	25.2
	OS-ABA	80.4	29.6	61.1	19.9	49.0	26.8	49.7	**26.4**
MobNetv2	OP-ABA	80.2	**34.7**	60.2	**26.9**	44.6	**29.7**	44.7	**28.1**
	OS-ABA	80.2	31.9	60.2	24.0	44.6	22.5	44.7	18.7

4.2.4 Attacking Details

Intuitively, given a targeted tracker, we can attack it by blurring each frame through OP-ABA and OS-ABA during the online tracking process, as shown in Fig. 1.2. The attack could be white-box. The tracking model in Eq. (4.4) is the same as the targeted tracker, leading to high accuracy drop. It also could be black-box, also known as the transferability; that is, the tracking model Eq. (4.4) is different from the targeted one. OP-ABA is time-consuming and based on iterative optimization;

thus, we conduct OP-ABA every five frames while performing OS-ABA for all frames. In practice, we blur the search regions between two frames to accelerate the attacking speed. Specifically, at the frame t, we crop a search region centered at the detected object as the $\mathbf{I}_t$. At the same time, we crop a region from the previous frame at the same position as the $\mathbf{I}_{t-1}$. Then, we use the PWCNet [100] to calculate optical flow. We get the original response map with the targeted tracker and $\mathbf{I}_t$ if we employ the OP-ABA as the attack method. After that, we can conduct the OP-ABA or OS-ABA to generate the adversarial blurred frame. Regarding the OP-ABA, we set the iteration number to be ten, and the step sizes for updating $\mathcal{W}_t$ and $\mathcal{A}_t$ are set as 0.002 and 0.0002, respectively. The number of intermediate frames N is fixed as 17 for both OP-ABA and OS-ABA.

4.3 Experiments

We design experiments to investigate three aspects: *First*, we validate the effectiveness of our two methods against state-of-the-art trackers on four public tracking benchmarks in Sec. 4.3.2. *Second*, we design ablation experiments to validate the influences of $\mathcal{A}_t$ and $\mathcal{W}_t$ in Sec. 4.3.3. *Third*, we compare our method with the state-of-the-art tracking attacks on their transferability and frame quality in Sec. 4.3.4.

4.3.1 Setups

Datasets. We evaluated adversarial blur attack on four popular datasets, *i.e.*, VOT2018 [65], OTB100 [118], UAV123 [85], and LaSOT [32]. VOT2018 and OTB100 have widely used datasets containing 100 and 60 videos, respectively. LaSOT is a recent large-scale tracking benchmark, which contains 280 videos. UAV123 [85] focuses on tracking the object captured by an uncrewed aerial vehicle's camera, including 123 videos.

Tracking models. We conduct attack against state-of-the-art trackers including SiamRPN++

[69] with ResNet50 [53] and MobileNetv2 [59], DiMP [4] with ResNet50 and ResNet18, and KYS [6]. Specifically, we validate the white-box attack with OP-ABA and OS-ABA against SiamRPN++ [69] with ResNet50 [53] and MobileNetv2 in Sec. 4.3.2 where the targeted tracker's model itself guides the motion-blurred frames. We choose SiamRPN++ [69] since it is a classic tracker for Siamese network-based methods [106; 70; 29; 3; 46] which achieves excellent tracking accuracy and real-time tracking speed. We also conduct transferability experiments using the motion blur crafted from SiamRPN++ with ResNet50 to attack other trackers.

Metrics. In terms of the OTB100, UAV123, and LaSOT datasets, we follow their common setups and use one pass evaluation (OPE) that contains two metrics *success rate* and *precision*. The former is based on the intersection over union (IoU) between the ground truth bounding box and predicted one for all frames. In contrast, the latter is based on the center location error (CLE) between the ground truth and prediction. Please refer to [118] for details. To evaluate the attack's capability, we use the drop of success rate and precision for different attacks, denoted as Succ. Drop and Prec. Drop. The higher dots mean more effective attacking. In terms of VOT2018, it restarts trackers when the object is lost. Expected average overlap (EAO) [66] is the main criterion, evaluating both accuracy and robustness. Like Succ. Drop, we use the drop of EAO (*i.e.*, EAO Drop) for evaluating attacks. Compared with other additive noise-based attacks, we use the BRISQUE [84] as the image quality assessment. An invasion is desired to produce adversarial examples that are natural and can fool trackers. BRISQUE is a common metric to evaluate the naturalness of images, and a smaller BRISQUE means a more natural image.

Baselines. There are several tracking attacks, including cooling-shrinking attack (CSA) [124], SPARK [51], One-shot-based attack [13], and PAT [114]. Among them, CSA and SPARK have released their code. We select CSA and SPARK as the baselines.

4.3.2 Validation Results

Attacking results. We attack two SiamRPN++ trackers using ResNet50 and MobileNetv2 as the backbone. The attack results on the four public datasets are presented in Table 4.1 and 4.2, respectively. We observe that: ❶ Both OP-ABA and OS-ABA significantly reduce the success rate and precision of the two targeted trackers on all benchmarks. Specifically, on the OTB100 dataset, OP-ABA makes the precision and success rate of SiamRPN++ with ResNet50 reduce 41.7 and 31.2, respectively, almost fifty percent of the original scores. These results demonstrate that the proposed attacks can fool the state-of-the-art trackers effectively. ❷ Compared with OS-ABA, OP-ABA achieves higher precision drop since it targets attacks to a specific position during each optimization. At the same time, OS-ABA generates a generally blurred image to make objects invisible to trackers. In general, all the results indicate the effectiveness of OP-ABA and OS-ABA in misleading the tracking models by adversarial blur attacks. ❸ Comparing the performance drop of SiamRPN++ (ResNet50) with SiamRPN++ (MobileNetv2), we observe that the former usually has relatively less precision or success rate drop under the same attack, hinting that the lighter model is fooled more easily. ❹ According to the visualization results shown in Fig. 4.3, we see that both methods can generate visually nature blurred frames that mislead the SiamRPN++. In

Table 4.3: Speed and time cost of three attacks and SiamRPN++ with the ResNet50 and MobileNetv2.

SiamRPN++	Attackers	Org. FPS	Attack time (ms) per frame ↓	Attack FPS ↑
ResNet50	OP-ABA	70.25	661.90	6.79
	OS-ABA	70.25	**42.97**	**17.62**
MobNetv2	OP-ABA	107.62	508.30	8.79
	OS-ABA	107.62	**40.88**	**19.96**

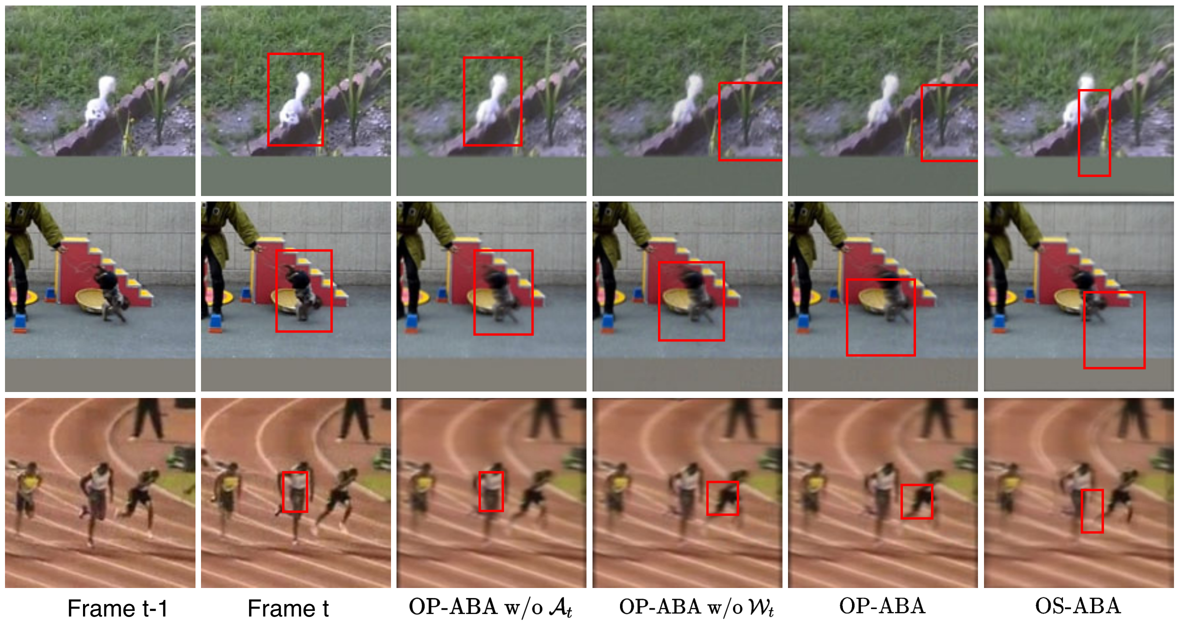

Figure 4.3: Three visualization results of OP-ABA w/o $\mathcal{A}_t$, OP-ABA w/o $\mathcal{W}_t$, OP-ABA, and OS-ABA against SiamRPN++ (ResNet50). The corresponding tracking results are shown with **red** bounding boxes.

general, OP-ABA contains some artifacts but can mislead the tracker more effectively than OS-ABA. In contrast, OS-ABA always generates more realistic motion blur than OP-ABA in all three cases.

Speed analysis. We test the time cost of OP-ABA and OS-ABA on the OTB100 and report the FPS of the SiamRPN++ trackers before and after attacking. As presented in Table 4.3, we observe that OP-ABA would significantly slow down the tracking speed. For example, due to online optimization, OP-ABA reduces the speed of SiamRPN++ with ResNet-50 from 63 FPS to 6.79 PFS. Thanks to the one-step optimization via JAMANet in Sec. 4.2.3, OS-ABA is almost ten times faster than OP-ABA according to the average attack time per frame. Consequently, OS-ABA achieved near real-time speed, *e.g.*, 17.62 FPS and 20.00 FPS, in attacking SiamRPN++ (ResNet50) and SiamRPN++ (MobileNetv2). Regarding the FPS after attacking, OS-ABA is also about three times faster than OP-ABA.

4.3.3 Ablation Study

Table 4.4: Effects of $\mathcal{W}_t$ and $\mathcal{A}_t$ to OP-ABA and OS-ABA by attacking SiamRPN++ (ResNet50) on OTB100. The best results are highlighted by **red** color.

Attackers	Succ. Rate	Succ. Drop ↑	Prec.	Prec. Drop ↑
Original	66.5	0.0	87.8	0.0
Norm-Blur	65.3	1.2	86.2	1.6
OP-ABA w/o $\mathcal{A}_t$	51.5	15.0	67.6	20.2
OP-ABA w/o $\mathcal{W}_t$	40.9	25.6	53.4	34.4
OP-ABA	35,3	**31.2**	46.1	**41.7**
OS-ABA w/o $\mathcal{A}_t$	61.0	5.5	80.8	7.0
OS-ABA w/o $\mathcal{W}_t$	41.6	24.9	58.3	29.5
OS-ABA	38.4	**28.1**	55.3	**32.5**

In this section, we discuss the influence of $\mathcal{W}_t$ and $\mathcal{A}_t$ to OP-ABA and OS-ABA by constructing two variants of them to attack SiamRPN++ (ResNet50) tracker on OTB100 dataset. Specifically, for both attacks, we only tune $\mathcal{A}_t$ and fix $\mathcal{W}_t$ as $\mathcal{W}_{\text{norm}}$, thus we get two variants OP-ABA w/o $\mathcal{W}_t$ and OS-ABA w/o $\mathcal{W}_t$. Similarly, we replace $\mathcal{A}_t$ with $\mathcal{A}_{\text{norm}}$ and adversarially tune $\mathcal{W}_t$, thus we get OP-ABA w/o $\mathcal{A}_t$ and OS-ABA w/o $\mathcal{A}_t$, respectively. Moreover, we build the' Norm-Blur' attack to demonstrate that the adversarial blur reduces the performance. It synthesizes the motion blur with $\mathcal{A}_{\text{norm}}$ and $\mathcal{W}_{\text{norm}}$, representing the usual blur that may appear in the real world.

We summarize the results in Table 4.4 and Fig. 4.3 and have the following observations: ❶ When we fix the $\mathcal{W}_t$ or $\mathcal{A}_t$ for OP-ABA and OS-ABA, the success rate and precision drops decrease significantly, demonstrating that tuning both motion patterns (*i.e.*, $\mathcal{W}_t$) and accumulation strategy ($\mathcal{A}_t$) can benefit the adversarial blur attack. ❷ According to the variance of the performance drop, we see that tuning the accumulation strategy ($\mathcal{A}_t$) contributes more to effective attacks. For example, without tuning $\mathcal{A}_t$, the success rate drops from 28.1 and 31.2 to 5.5 and 15.0 for OS-ABA and OP-ABA, respectively. ❸ SiamRPN++ are robust to the Norm-Blur with slight success rate and precision drops. In contrast, the adversarial blur causes a significant performance drop, demonstrating the adversarial blur does pose a threat to visual object tracking. ❹ According to the visualization results in Fig. 4.3, we have similar conclusions with the quantitative results in Table 4.4: OP-ABA w/o $\mathcal{A}_t$ can generate motion-blurred frames but have little influence on the prediction accuracy. Once we tune $\mathcal{A}_t$, the tracker can be fooled effectively, but some artifacts are also introduced.

4.3.4 Comparison with Other Attacks

In this section, we study the transferability of proposed attacks by comparing them with baseline attacks, *i.e.*, CSA [124] and SPARK [51]. Specifically, for all compared attacks, we use SiamRPN++ (ResNet50) as the guidance to perform optimization or training. For example, we set ϕ_{θ_t} in the objective function of OP-ABA (*i.e.*, Eq. (4.4)) as the model of SiamRPN++ (ResNet50). We report the precision drop after attacking in Table 4.5 and the BRISQUE as the image quality assessment for generated adversarial frames.

As shown in Table 4.5, we observe: ❶ Our methods, *i.e.*, OP-ABA and OS-ABA, achieve the best and second-best transferability (*i.e.*, higher precision drop) against DiMP50, DiMP18 [4], and KYS [6], hinting that our methods are more practical for black-box attacking. ❷ According to BRISQUE results, the adversarially blurred frames have smaller values than other adversarial examples, hinting that our methods can generate more natural frames since motion blur is an expected degradation in the real world.

Table 4.5: Comparison results on transferability. Specifically, we use the adversarial examples crafted from SiamRPN++ (ResNet50) to attack four state-of-the-art trackers including SiamRPN++ (MobileNetv2) [69], DiMP50 [4], DiMP18 [4], and KYS [6] on OTB100. We also calculate the average BRISQUE values of all adversarial examples.

Trackers	SiamRPN++ (MobNetv2)	DiMP50	DiMP18	KYS	BRISQUE ↓
Org. Prec.	86.4	89.2	87.1	89.5	20.15
CSA	0.2	3.4	2.7	0.8	33.63
SPARK	**0.9**	2.0	1.0	0.9	24.78
OP-ABA	**2.5**	**6.6**	**10.3**	**7.9**	**21.39**
OS-ABA	0.2	**10.7**	**11.2**	**12.3**	**22.94**

4.3.5 Conclusion

In this chapter, we proposed a novel adversarial attack against visual object tracking, *i.e.*, adversarial blur attack (ABA), considering the effects of motion blur instead of noise against the state-of-the-art trackers. We first identified the motion blur synthesizing process during tracking, based on which we proposed the optimization-based ABA (OP-ABA). This method fools the trackers by iteratively optimizing a tracking-aware objective but causes heavy time costs. We further proposed the one-step ABA by training a novel-designed network to predict blur parameters in a one-step way. The attacking results on four public datasets, the visualization results, and the comparison results demonstrated the effectiveness and advantages of our methods. This work not only reveals the potential threat of motion blurs against trackers but also could work as a new way to evaluate the motion-blur robustness of trackers in the future.

5 | Misguiding the Visual Tracker via Backdoor Learning

5.1 Introduction

While the previous chapter addressed deficiencies in the model during testing, there are also risks of encountering attacks during the training process. One such method of attack is the backdoor attack, where a backdoor is injected during model training. As a result, the model is susceptible to targeted attacks during testing via a trigger, which can be challenging for users to detect. In this chapter, we focus on the backdoor attack in the context of visual object tracking. We present a novel backdoor attack approach that targets the model's training process and leverages the inherent characteristics of visual object tracking to embed a backdoor. Through this attack, an adversary can effectively manipulate the tracking process by introducing a trigger that can be activated at any time, leading to tracking a different object or the complete failure of the tracking process. This can have significant consequences in real-world scenarios, such as in autonomous vehicles or surveillance systems, where the reliability and accuracy of the tracking model are critical. By studying the backdoor attack in the context of visual object tracking, we hope to raise awareness of this threat and develop effective countermeasures to mitigate the risk of such attacks on visual tracking systems.

Therefore, we propose a novel and effective targeted backdoor attack approach TAT against VOT. In particular, the TAT framework comprises two versions: TAT-BA (Basic Attack) and TAT-DA (Advanced Attack). The TAT-BA version can execute efficient and covert backdoor attacks against VOT trackers. In contrast, the TAT-DA version can evade two prominent defense techniques in Sect.5.3. We conducted an extensive large-scale experimental evaluation to validate the effectiveness and stealthiness of the TAT framework. The results of our experiments not only confirm the robustness and efficacy of TAT and highlight the successful execution of TAT-BA in real-world scenarios. Furthermore, we assessed the capabilities of TAT-DA in countering commonly employed defense techniques in Sect.5.4.

5.2 Problem Statement

This section presents the problem statement of targeted backdoor attacks in VOT tasks. We introduce *Siamese-based networks*, frequently used as trackers in VOT tasks. Next, we describe the problem setting of targeted backdoor attacks, including the objectives and desired properties of these attacks.

Let $V = \{X_i\}_{i=1}^{N}$ be a video with N frames, where X_i is the i-th frame. We denote by $B_{gt} = \{b_{gt}^{i} \in \mathbb{R}^{4\times1}\}_{i=1}^{N}$ a sequence of bounding boxes, each element b_{gt}^{i} denoting the ground-truth bounding box of the tracking target. The objective of VOT is to make a sequence $B_{pr} = \{b_{pr}^{i} \in \mathbb{R}^{4\times1}\}_{i=1}^{N}$ of predictions, such that the distance between B_{pr} and B_{gt} is minimal. There are various distance measures in existing datasets [117; 86; 61; 32], and the measure selection depends on the documentation of concrete datasets.

Siamese-Based Networks. Siamese-based networks [123; 68; 3; 120] are among the most commonly used methods in VOT tasks. As shown in Fig. 5.1, these networks take an image $T \in \mathbb{R}^{3\times H_t\times W_t}$ as a template and an image $S \in \mathbb{R}^{3\times H_s\times W_s}$ as a search region (where typically $H_s > H_t$

and $W_s > W_t$). They are trained to locate the tracking target indicated by T within S and output a bounding box b_{pr} that shows the target's position in S.

To achieve this, Siamese-based networks generally consist of three CNN models: a backbone $\varphi(\cdot)$, a classification branch $f_\theta^{cls}(\cdot)$, and a regression branch $f_\theta^{reg}(\cdot)$. The network's workflow is described as follows:

- the backbone respectively extracts a feature $F_t = \varphi(T)$ from the template T and a feature $F_s = \varphi(S)$ from the search region S;
- the classification branch infers the similarity between F_t and F_s to generate a response map $M_p = f_\theta^{cls}(F_t, F_s) \in \mathbb{R}^{H_m \times W_m}$, which is a $H_m \times W_m$ matrix that, given a location (x, y), returns a real value $M_p(x, y)$ that indicates the likelihood of (x, y) being the center of the tracking target in S;
- to robustify the network against the surrounding interference, a *Cosine Window Penalty (CWP)* [3] is often introduced to punish the edge of the map M_p, which derives a new map M_f;
- lastly, the regression branch gives the final bounding box $b_{pr} = f_\theta^{reg}(x', y')$ as the output of the whole network, where $(x,' y') = \underset{(x,y)}{\text{argmax}} M_f(x, y)$ is the location of the maximum value in the map M_f.

To summarize, the workflow of the Siamese-based network can be formalized as $b_{pr} = f_\theta(T, S)$, where f_θ is a function consisting of all the procedures described above and is parametrized by θ. The network is trained by tuning the parameters θ to minimize the following loss function:

$$L_{Track} = L(f_\theta(T, S), B_{gt}) \tag{5.1}$$

Targeted Backdoor Attack in VOT: Attacking Capabilities and Goals. In targeted backdoor attacks on VOT tasks, we assume that the adversary has complete control over the training process, including the ability to modify the training data, model structure, and training loss, and

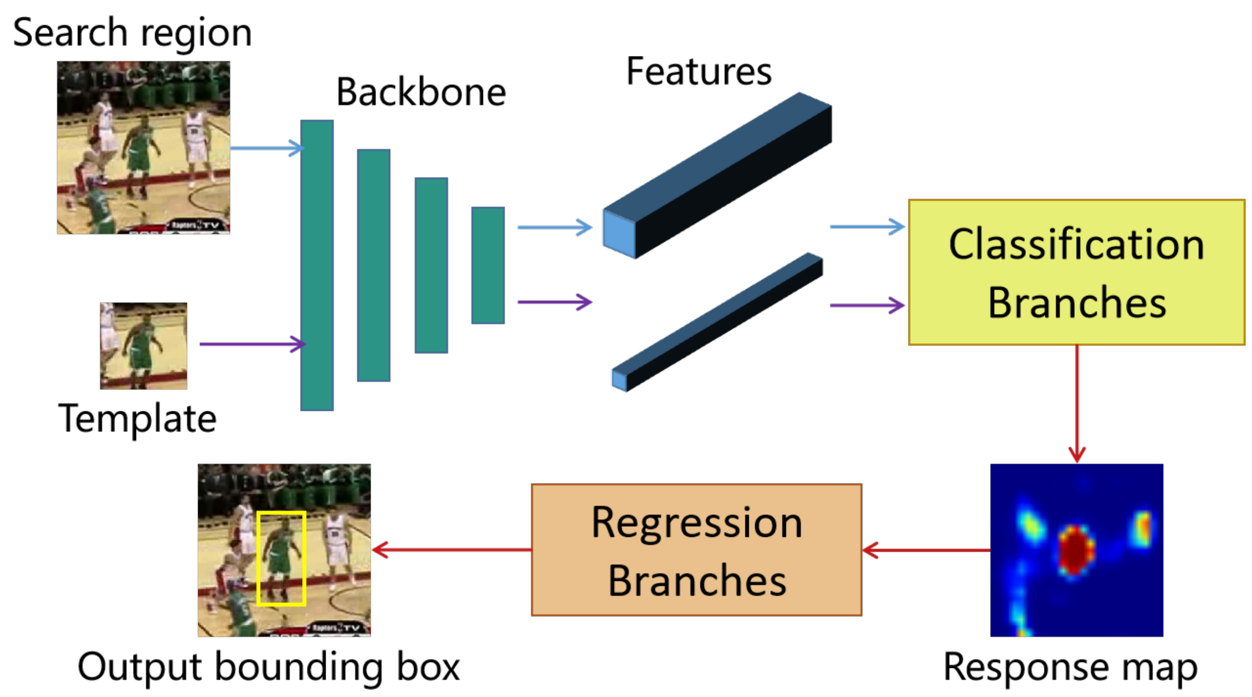

Figure 5.1: Siamese-based network.

can output a backdoored model.

The goal of the adversary in targeted backdoor attacks is to mislead the tracker model such that it tracks a trigger (an image injected into the template and search region) instead of the original tracking target indicated by the template. The attack is designed to satisfy two key properties:

- **Effectiveness:** the tracker should follow the trigger once it appears in the search region rather than the original tracking target;
- **Stealthiness:** the tracker should be sufficiently stealthy to counter some defenses; for example, it should not be too visible to human eyes, and in the case when the search region is clean, it should be able to track the original tracking target, etc.

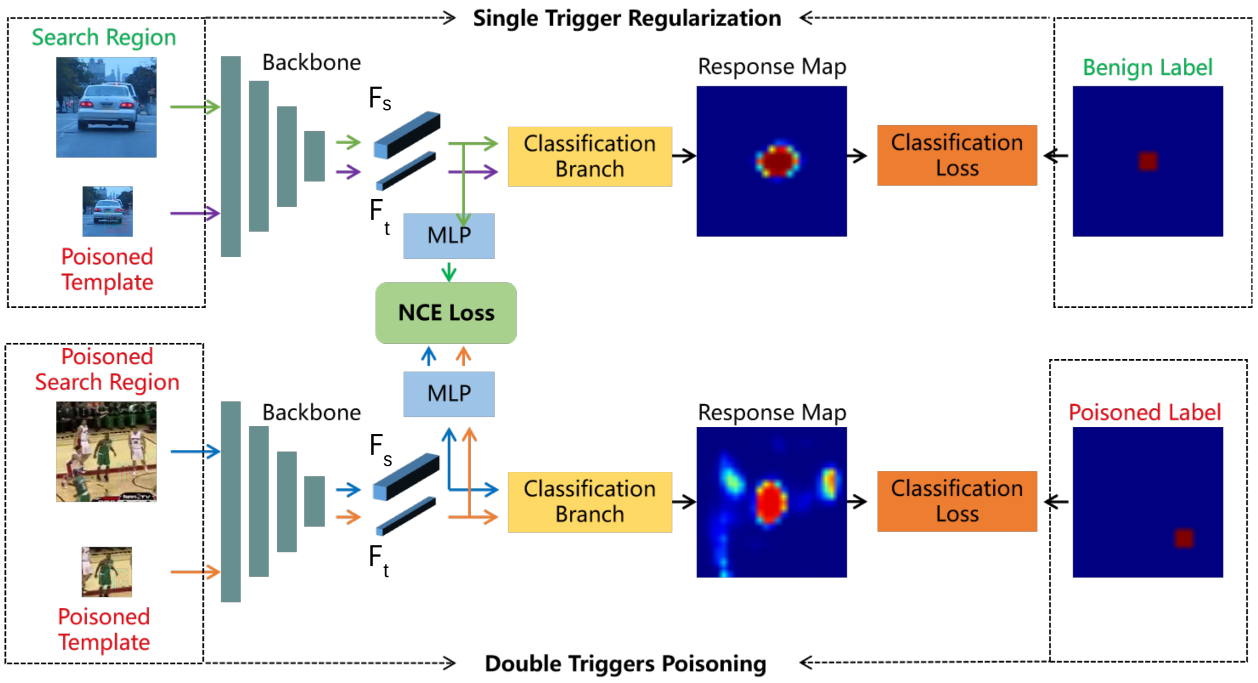

Figure 5.2: The injecting backdoor process of TAT during training.

5.3 The Proposed Approach

This section presents our proposed approach for backdoor attacks on VOT, called the TAT framework. We introduce a basic version of TAT, TAT-BA, in Sect.5.3.1, which uses the double trigger poisoning technique to ensure effectiveness and is further enhanced by adopting the NCE loss and STR strategy. We then introduce a defense-aware version of TAT, TAT-DA, in Sect.5.3.3, which can counter two commonly used backdoor defense techniques.

5.3.1 TAT-BA: the basic TAT

In image classification tasks, backdoor attacks typically involve using a trigger (i.e., an image used to poison the target being attacked) to mislead the classifier into producing incorrect classification results. In contrast, backdoor attacks in VOT are designed to deceive the tracker in a sequence of frames. Specifically, the aim is to make the tracker follow the trigger in the search region across the frame sequence rather than tracking the template. To achieve this goal, the same trigger (e.g., BadNet [40]) must be added to both the template and the search region.

To accomplish this, we propose a technique called *double trigger poisoning (DTP)*, which involves adding the same trigger to both the template and the search region.

Double Trigger Poisoning (DTP). DTP involves two phases: a training set poisoning phase and a training phase. In the first phase, we poison a portion of the training data by injecting the triggers and changing their labels. Specifically, we select a portion $\gamma\%$ of the training data and poison them by patching the trigger respectively at the location (x_s, y_s) in the search region and the central location in the template.

The ground truth labels are stored in the response map M_p, i.e., $M_p(x, y) \in \{0, 1\}$, where the value 1 indicates that the bounding box significantly overlaps the target, and the value 0 indicates little overlap. To modify the labels, we falsify the values in M_p to produce a new map $\hat{M}$, which is defined as follows:

$$\hat{M}(x, y) = \begin{cases} 1 & \text{if } (x, y) = (x_m, y_m) \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

where,

$$\begin{aligned} x_m &= \text{round}\left(\left(x_s - \frac{W_t}{4}\right) \cdot \frac{W_m}{W_s - \frac{W_t}{2}}\right) \\ y_m &= \text{round}\left(\left(y_s - \frac{H_t}{4}\right) \cdot \frac{H_m}{H_s - \frac{H_t}{2}}\right) \end{aligned} \tag{5.3}$$

Eq. 5.3 maps the location where the trigger is added in the search region to the response map, and Eq. 5.2 modifies the values in that location to 1. By doing that, we tamper with the training data and their labels. This process is also illustrated by the bottom half part of Fig. 5.2, where DTP is used to poison both the search region and the template, and the label is modified accordingly.

We train the VOT model using the poisoned training dataset in the second phase. For the poisoned data, the training loss function is formulated as follows:

$$L_{DTP} = \alpha \cdot L_{cls}(f_{\theta}^{cls}(\tilde{T}, \tilde{S}), \hat{M}) \tag{5.4}$$

In L_{DTP}, we use $\tilde{T}$ and $\tilde{S}$ to represent the poisoned template and the poisoned search region, respectively. The total loss function is the sum of L_{DTP} for the poisoned data and L_{Track} in Eq. 5.1 for the remaining data.

So far, we have introduced DTP, a backdoor attacking method based on patching the same trigger to both the template and the search region. However, by using it individually for backdoor attacking, there are apparent limitations in the attacking performances in the following aspects (which is also supported by our empirical study in Sect. 5.4.3):

Pt (1) The adopted trigger that follows the pattern of BadNet [40] is visible to humans and, as a result, it is not stealthy enough to resist the potential backdoor defense technique;

Pt (2) In the case when only the template is poisoned (while the search region remains benign) in the testing phase, although the tracker is expected to track the original object in the template, it may actually not behave in that way since there is no training data that guide in that direction.

To improve the performance of backdoor attacking, we propose two techniques that respectively address the two problems existing in DTP; namely, we propose to use the *noise contrastive estimation loss* to solve Pt 1, and we propose to use the *single trigger regularization* method to solve Pt 2.

Noise Contrastive Estimation (NCE) Loss. To improve the stealthiness of the attack (as mentioned in Pt 1), we adopt the state-of-the-art trigger generation techniques in the backdoor attacks for image classification [74; 90], in which they generate various triggers by using the *U-net* architecture [96]. In these works, they use a *cycle-GAN* $g(\cdot)$, introduced from [135], as a generator to produce dynamic triggers. In our context, for the template and the search region, we follow the steps below, respectively, to obtain the poisoned images:

- first, we select a random location from the template and crop a sub-image $I_{tc} \in \mathbb{R}^{3\times H_c\times W_c}$; similarly, we obtain a sub-image $I_{sc} \in \mathbb{R}^{3\times H_c\times W_c}$ from the search region;
- then, we poison the sub-images I_{tc} and I_{sc} to obtain two triggers $\hat{I}_{tc}$ and $\hat{I}_{sc}$, as follows:

$$\hat{I}_{tc} = (1-\mu)\cdot I_{tc} + \mu\cdot g(I_{tc}) \tag{5.5}$$

$$\hat{I}_{sc} = (1-\mu)\cdot I_{sc} + \mu\cdot g(I_{sc}) \tag{5.6}$$

  where μ is a hyperparameter that decides the stealthiness: the smaller μ is, the more stealthy the attack is; however, μ cannot be too small since that can diminish the trigger and thus harm the training. In this work, we set $\mu = 0.2$ for attacking SiamRPN++ and SaimFC++, $\mu = 0.1$ for attacking STARK.
- lastly, we paste I_{tc} and I_{sc}, respectively, to the original template and the original search region, which results in the poisoned template $\hat{T}$ and the poisoned search region $\hat{S}$, respectively.

Compared to the original DTP, this cycle-GAN-based method can generate less visible triggers and thus improve the stealthiness of the attack. However, a problem also arises because the poisoning effects imposed by the triggers to the template and the search region are relatively weak. The features of the stimuli, exposed by the backbone, in the template feature F_t and the search region feature F_s, are very different. Consequently, this goes against the goal of backdoor attacking, i.e., the tracker cannot be fooled to track the trigger in the search region. To solve this problem, we propose to use the *noise contrastive estimation (NCE) loss* as a complement to the original loss function.

NCE loss is a technique widely applied in *contrastive learning* [104; 52; 72]. In our context, NCE loss is adopted for two purposes: first, it is used to drive the features of the poisoned template and the poisoned search region closer, in line with the attacking goal that the tracker can track the trigger in the poisoned search region; second, it also separates the feature of poisoned template away from the features of unrelated benign search regions, to avoid training problems such as *model collapse* [63].

Technically, NCE loss is integrated into our framework in the way illustrated in Fig. 5.2. Let (F_t^i, F_s^i) be a pair of features extracted from a poisoned training data i, including a poisoned template and a poisoned search region, and (F_t^j, F_s^j) be a pair of features extracted from an unrelated training data j including a poisoned template and a clean search region. For both of the data i and j, we project their features to a new feature space by introducing a new branch of *multi-layer perceptron (MLP)* that consists of two fully-connected layers (indicated by the blue boxes of MLP in Fig. 5.2). As a result, the NCE loss is formulated as follows:

$$L_{NCE} = -\log\left(\frac{\exp(\pi(F_t^i)\cdot\pi(F_s^i))}{\exp(\pi(F_t^i)\cdot\pi(F_s^j)) + \exp(\pi(F_t^i)\cdot\pi(F_s^i))}\right) \tag{5.7}$$

where π denotes the projection by MLP. Intuitively, L_{NCE} tends to make the features of the poisoned search region and template similar while pulling away the distance between the features of a poisoned search region and a benign unrelated one, such that the tracker can be more focused on the differences in these samples (i.e., whether or not a trigger is added).

Single Trigger Regularization (STR). To ensure that the tracker can still follow the object when the search region remains benign (as mentioned in Pt 2), we propose a method named *single trigger regularization (STR)* as a complement loss aiming to train the tracker such that it can behave as expected in that specific occasion. Essentially, STR is designed to prevent the tracker from focusing only on the trigger. Concretely, given a subset of the training data (i.e., a search region, a template, and their label), we only poison their templates, but we keep the search regions and the labels unchanged; the corresponding loss function for these data is thus expressed as follows:

$$L_{STR} \;=\; L(f_\theta(\hat{T}_j, S_j), B_{gt}) \tag{5.8}$$

Intuitively, STR makes the tracker consider the features of the real target in the poisoned template rather than the feature of the trigger only. In conjunction with DTP, the backbone allows triggers to express more similar features (the higher value in the response map) while still outputting the features of the real object.

The Overall Approach. Now we present our overall approach of TAT-BA. The description of the poisoning data in the training phase is shown in Fig. 5.2. After the integration of NCE loss and STR, the loss function used for training a tracker is formulated as follows:

$$L_{total} \;=\; \alpha \cdot L_{DTP} + \beta \cdot L_{STR} + \delta \cdot L_{NCE} + L_{Track} \tag{5.9}$$

During training, for each mini-batch, we randomly choose $\frac{\gamma}{2}$% unrelated training samples for the DTP and the STR pipeline (as illustrated in Fig. 5.2), respectively. Note that the DTP poisons both the template and the search region, while the STR poisons only the template. Then we calculate the NCE loss based on the feature tensors of two branches. Finally, we update the parameters of the tracker and the trigger generator $g(\cdot)$.

5.3.2 TAT-EOT:

In the physical world, attacking the trackers is hard to be as effective as in the digital setting because of lighting changes, trigger distortion, fast-moving targets, and Gaussian noise. Inspired by *Expectation Over Transformation (EOT)* [1], which can synthesize adversarial examples that are still robust in practical scenarios, we follow their ideas to improve the TAT. Specifically, we develop TAT-EOT by applying various transformations, including rotation, brightness variation, Gaussian blur, and Gaussian noise, on poisoned search images during training to improve the attacking robustness of TAT. In Sect. 5.4.5, we compare the performance of TAT-EOT with TAT-BA.

5.3.3 TAT-DA: Defense-Aware TAT

With the development of backdoor attacks, various defensive techniques are increasingly proposed as a countermeasure. In Sect. 2.6, we have reviewed the typical defense schemes, namely, the defense based on poisoned data detection and the protection based on backdoor removal. In this section, we extend the TAT-BA and propose the defense-aware technique TAT-DA, based on two representative defense techniques, namely, *STRIP* [37] that follows the line of poisoned data detection, and *Fine-Pruning* [80] that follows the line of backdoor removal.

STRIP-Aware Attack. In the original work of STRIP [37], many empirical studies are performed, which conclude that adding benign test data a to poisoned data b will not change the incorrect output $f_p(a+b)$ of the poisoned model, i.e., $f_p(b) = f_p(a+b)$; in contrast, if the model f is clean, despite whether b is clean, it normally holds that $f(b) \neq f(a+b)$. This difference can be used to judge whether a model is poisoned. Moreover, this technique can also be used in the context of VOT to detect the existence of backdoor attacks, including our TAT-BA; If $f_{\theta}^{cls}(\hat{S}, \hat{T}) = f_{\theta}^{cls}(\hat{S}+S_{ram}, \hat{T})$, where S_{ram} is a clean unrelated search region, the tracker has been injected with a backdoor. Our experiments in Sect. 5.4.6.2 empirically confirm this.

To counter this defense, we propose a simple yet effective STRIP-aware attack method. Given a training sample $((S_i, T_i), M_i)$, after patching the triggers to the search region S_i and the template T_i, we mix an irrelevant search region S_j with S_i, i.e., $((\hat{S}_i+S_j, \hat{T}_i), M_i)$. Then, the label of this mixed sample keeps the same as the original one except for the location of the trigger's response:

$$\hat{M}(x,y) = \begin{cases} 0 & \text{if } (x,y) = (x_m, y_m) \\ M(x,y) & \text{otherwise} \end{cases} \tag{5.10}$$

where y_m and x_m are calculated as in Eq. 5.3 with (x_s, y_s) being the position of the trigger in S_i, and $M(x,y)$ gives the label in the original map of S_i. Intuitively, to have $f_\theta^{cls}(\hat{S}, \hat{T}) \neq f_\theta^{cls}(\hat{S} + S_{ram}, \hat{T})$, we expect the trigger not to be activated in the presence of a strong noise such as S_j. To achieve this, the position the trigger should have activated in the response map will be set to 0, so the trigger is deactivated.

Pruning-Aware Attack. The Fine-Pruning method [80] aims to prune the unimportant channels which have a low response for clean samples but are activated by the triggers in the tracker model. If these unimportant channels are pruned, backdoor attacks will no longer be effective. Our experiments in Sect. 5.4.6.2 show that TAT-BA cannot escape such a backdoor defense.

To resist Fine-Pruning, we need to inject the triggers into similar channels of benign data, so we propose the Pruning-Aware TAT (TAT +PA). We apply a query mechanism: For each mini-batch of benign data, we first use them to prune 50% unimportant channels of the last layer based on the activation ranking of benign samples and then poison the data as done by TAT-BA to train the model. Intuitively, by applying our method, the poisoned samples and benign ones will activate similar channels such that Fine-Pruning cannot prune the channels activated by the triggers.

5.4 Experimental Evaluation

This section presents the experimental evaluation results on the performance of TAT. We first introduce the experiment settings in Sect. 5.4.1, and then in the remainder of this section, we show our experimental results, which aim to investigate the performance of TAT from the following aspects: (i) we validate the effectiveness of TAT-BA against two Siamese-based trackers on two public VOT benchmarks in Sect. 5.4.2; (ii) we conduct ablation experiments in Sect. 5.4.3 to validate the influences of DTP, NCE Loss, and STR introduced in Sect. 5.3.1; (iii) to illustrate the potential harm, we compare other backdoor attacks and adversarial attacks on VOT in Sect. 5.4.4. (iv) we demonstrate the robustness of the attack in real-world scenarios in Sect. 5.4.5; (v) we validate the stealthiness of the attack to evade the classic backdoor defense in Sect. 5.4.6.

5.4.1 Experiment Settings

Datasets and Models. We conduct TAT-BA against tracking model SiamRPN++ [68], and SiamFC++ [120] since they are classic Siamese-based networks and they have shown their high performance on various datasets. In particular, STARK [123] is a transformer-based tracker which is state-of-the-art in various datasets. We also attack it to prove the generalizability of TAT-BA. We select 4 representative datasets, namely, OTB100 [117], UAV123 [86], GOT10K [61] and LaSOT [32], to validate the effectiveness of TAT-BA. OTB100 is a universal tracking dataset with 100 videos. UAV123 focuses on moving objects with 123 videos. GOT10K gives 180 sequences for bounding box regression testing. LaSOT is a large-scale dataset including 280 videos.

Metrics. The VOT task aims to train a tracker that can predict the bounding boxes in a sequence of video frames as close to the real ones. Various datasets [117; 86; 61; 32] have different testing preferences. Based on the requirements of benchmarks, we evaluate the performance of trackers using the four metrics: (i) **Prec** shows the location precision indicated if the distance between the predicted bounding box and the ground truth is less than 20 pixels in the images; (ii) **AUC** represents the area under the success rate curve, which is used to measure the overlap ratio between the predicted box and the ground-truth one; (iii) **SR50** reflects the track success rate when the overlaps exceed a threshold of 0.5; (iv) **nPrec** is the normalized precision as defined in

[93].

To avoid misunderstandings, we use **AUC** and **Prec** to express the similarity between the predicted box and the ground-truth one in the datasets; we use **AUC** and **Prec** to reflect the similarity between the predicted box and the trigger. Similarly, we use green fonts for other metrics to denote the metric w.r.t. the predicted box and the ground truth and use red fonts to denote the metric w.r.t. the predicted box and the trigger.

Baseline Approaches. To show the effectiveness of our TAT-BA, we train another tracker with benign samples only as a baseline, and we test the baseline approach and TAT-BA with poisoned and benign data to see their effectiveness and stealthiness.

Attack Setup. In the training phase, we poison two samples respectively, as shown in Fig. 5.2, per 20 training samples. In the testing phase, we randomly select a position 40 pixels away from the center of the search region to inject the trigger. There are two reasons for this design: (i) The target is most likely to be in the center of the tracking task, and the trigger is not allowed to cover the target. (ii) If the trigger is too far from the target, it is challenging to suppress its activation in the response map as depicted in Fig. 5.3.

We consider two attack configurations, namely, *AT* and *ATS*. The former means adding only a trigger to the template, while the latter means simultaneously adding the trigger to the template and search region. On the template, the trigger is added at the bottom right corner. The width and height of the trigger are 64 pixels in both the search region and the template.

For stealthiness, we compare the performance of the poisoned tracker and benign one with benign data. The higher the green metrics value of TAT-BA tracker, the less TAT-BA hurts the model performance. In effectiveness, we compare the trackers' ability to track triggers. The higher the red metrics value of TAT-BA tracker, the stronger TAT-BA's ability to target attack trackers.

5.4.2 Effectiveness of TAT-BA

Table 5.1 shows the results of the attacks by TAT-BA to SiamRPN++, SiamFC++, and STARK on OTB100, UAV123, GOT10K, and LaSOT, respectively. The results deliver information in two aspects:

- Stealthiness of the attack: compared with benign training, our TAT-BA achieves a similar score when tested in clean data (Benign), e.g., TAT-BA based on SiamFC++ is 0.3% less AUC than baseline. Moreover, TAT-BA, with only adding a trigger on the template (AT), maintains acceptable AUC and precision in benign testing, e.g., SiamFC++ declines 2.2% of AUC on UAV123. This difference is hard to detect in practice.
- Effectiveness of the attack: TAT-BA achieves an attack success rate of at least 97.8% on OTB100 and 96.5% on UAV123 against SiamRPN++. In contrast, the real target has only about 3.1% probability of appearing in the location where the triggers are pasted, far below our attack accuracy. In terms of the model, SiamRPN++ is more robust than SiamFC++ since it employs three layers of features to generate three outputs and merge them, which can be seen as an ensemble model. This result may help to design more robust models.

In summary, the results indicate the stealthiness of TAT-BA in maintaining normal performance with benign data and the effectiveness of TAT-BA in attacking tracking models with triggers.

Visualization. To visualize the effectiveness of TAT-BA, we select some video frames in OTB100 as an example. As shown in Fig. 5.3, the search region and its response map are pasted into the bottom left corner of each frame. The real object is a pedestrian marked by a green box, and the tracker predicts the yellow box.

In the first frame, both the trigger and the pedestrian activate the map. Although the pedestrian has a more extensive response area, the trigger has a higher value. As a result, the predicted yellow box surrounds the trigger. This indicates that the tracker prioritizes the trigger, and TAT-

Table 5.1: Attacking and Benign results (%) of TAT-BA against SiamRPN++, SiamFC++ and STARK on OTB100, UAV123, GOT10K and LaSOT datasets.

Datasets↓	Train mode →	Benign				TAT-BA					
	Test mode →	Benign		ATS		Benign		AT		ATS	
OTB100	Metrics→	**AUC**	**Prec**	**AUC**	**Prec**	**AUC**	**Prec**	**AUC**	**Prec**	**AUC**	**Prec**
	SiamRPN++	65.1	85.7	12.5	15.4	64.6	85.2	61.4	81.6	75.8	97.8
	SiamFC++	66.5	89.0	23.7	18.9	66.2	88.7	58.9	81.7	56.1	99.8
	STARK	65.8	85.3	10.1	0.6	64.0	83.4	62.0	80.2	90.9	99.8
UAV123	Metrics→	**AUC**	**Prec**	**AUC**	**Prec**	**AUC**	**Prec**	**AUC**	**Prec**	**AUC**	**Prec**
	SiamRPN++	59.0	77.4	21.0	33.7	58.6	77.5	56.8	75.2	73.2	96.5
	SiamFC++	61.4	78.8	16.2	17.8	61.1	78.9	58.3	75.4	53.4	98.9
	STARK	63.0	83.0	10.2	0.7	62.1	82.2	62.0	81.8	89.5	99.2
LaSOT	Metric	**AUC**	**nPrec**	**AUC**	**nPrec**	**AUC**	**nPrec**	**AUC**	**nPrec**	**AUC**	**nPrec**
	SiamRPN++	50.1	57.9	23.3	15.6	49.4	57.8	47.0	54.5	71.7	91.0
	SiamFC++	54.2	57.1	21.6	19.3	53.0	56.5	47.3	54.4	78.3	99.9
	STARK	59.3	68.0	11.4	1.4	58.5	66.9	58.0	66.2	89.3	99.9
GOT10K	Metric	**AUC**	**SR50**	**AUC**	**SR50**	**AUC**	**SR50**	**AUC**	**SR50**	**AUC**	**SR50**
	SiamRPN++	67.4	78.1	11.4	8.1	65.6	77.6	63.2	75.5	64.9	87.5
	SiamFC++	74.8	87.8	14.2	2.9	74.1	87.4	70.0	82.7	89.1	98.5
	STARK	73.2	84.7	10.6	1.1	72.9	85.4	70.6	81.2	90.4	99.3

BA can control it more precisely. Thanks to the Cosine Window Penalty (CWP, recall in Sect. 5.2), which gives more weight to the central positions of the search region, the real object gradually shifts away from the center of the search region with no possibility of being recovered in the follow-up frames.

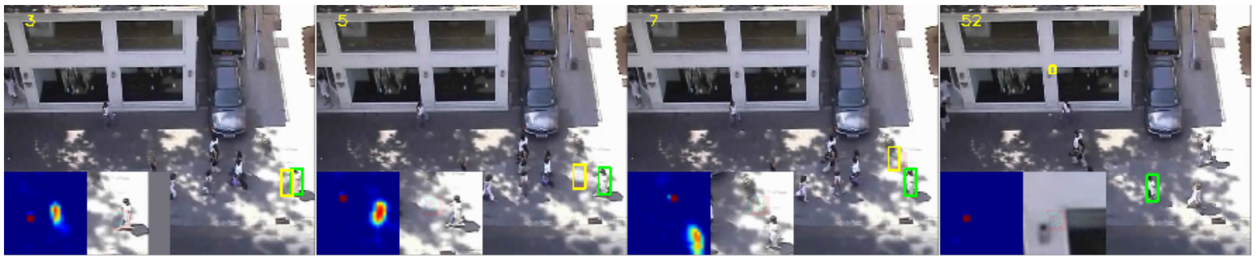

Figure 5.3: Some frames of "Crowds" in OTB100. The response map and search are pasted in the lower-left corner of each frame.

5.4.3 Ablation Studies

To understand the influence of DTP, NCE Loss, and STR introduced in Sect. 5.3.1, we assess the performances of TAT-BA against SiamRPN++, in the absence of the above mechanisms. The result is summarized in Table 5.2.

TAT-BA without Double Trigger Poisoning. As discussed in Sect. 5.3.1, DTP means adding trigger-pair to the template and the search region, respectively, for producing a strong match response compared to normal samples. This technique is the core factor of a successful targeted backdoor attack. In our experiments, we replace DTP by adding a single trigger to search region and evaluate the performance, namely TAT-BA w/o DTP in Table 5.2.

The 3.1% Prec of ATS, equivalent to *Random* in Table 5.1, reflects that there is no attacking ability of TAT-BA without DTP. This is because the tracker's output depends on the similarity of the search and template.

TAT-BA without NCE Loss. As described in Sect. 5.3.1, NCE loss brings two images with triggers closer while putting poisoned data away from a benign one. It contributes to the convergence of the model based on the low blending factor (e.g., $\mu = 0.2$ for SiamRPN++) of triggers. Therefore, TAT-BA only achieves 3.2% precision without it, corresponding to random sampling in Table 5.2.

TAT-BA without Single Trigger Regularization. The STR method prevents model overfitting

Table 5.2: Attacking and Benign (%) of TAT-BA without different training settings, e.g., STR, NCE loss, and DTP. The attacked tracker is SiamRPN++

Train mode	TAT-BA						TAT-BA w/o DTP					
Test mode	Benign		AT		ATS		Benign		AT		ATS	
Metrics	AUC	Prec	AUC	Prec	AUC	Prec	AUC	Prec	AUC	Prec	AUC	Prec
OTB100	64.6	85.2	61.4	81.6	75.8	97.8	64.1	84.2	62.4	83.8	6.1	3.1
Train mode	TAT-BA w/o NCE						TAT-BA w/o STR					
Test mode	Benign		AT		ATS		Benign		AT		ATS	
Metrics	AUC	Prec	AUC	Prec	AUC	Prec	AUC	Prec	AUC	Prec	AUC	Prec
OTB100	63.7	85.6	62.5	85.3	6.1	3.2	63.7	84.0	38.0	52.7	78.6	99.9

by setting one trigger on the template while keeping its label unchanged. This method requires the model not to cause excessive deviations in the latent feature space.

As shown in Table 5.2, when TAT-BA injects a trigger on the template without using STR, it experiences a 25.7% AUC drop compared to benign testing. This drop indicates that the attacked model considers the trigger a standard pair of templates and search region, and the triggered template cannot identify the target.

5.4.4 Comparison with Other Attacks

To show the destructiveness of TAT-BA, we compare the performance degradation of the attacked trackers with other adversarial attack and backdoor attack methods. We select the five works, including OSAA [13], SPARK [50], CSA [124], FSBA [77] and EAA [78], to compare the performance of attacks. All of these methods attack SiamRPN++ (ResNet50) on OTB100. In our experiment, TAT-BA just attacks the trackers on the first 10% frames for each video, while SPARK

Method	OSAA	SPARK	CSA	FSBA	EAA	TAT-BA
Success Drop	44.4	-	37.2	55.4	-	**56.6**
Precision Drop	57.7	29.8	44.3	74.6	40.8	**75.3**

Table 5.3: Attacking SiamRPN++ on OTB100.

and CSA attack every frame of OTB100, some of which are not targeted attacks.

The results can be seen in Table 5.3. We use the absolute value of the performance drop to express the effectiveness of the attack. We can see that, even though the frames being attacked by TAT-BA are less than that by other approaches, our approach TAT-BA still achieves the most significant score drop.

5.4.5 Real-World Attack

In this section, we apply TAT-BA against the SiamFC++ tracker in a real-world setting. We also compare the performance of TAT-EOT, described in Sect. 5.3.2, with TAT-BA on attacking the SiamRPN++ tracker.

We directly deploy the poisoned tracker trained in Sect.5.4.2 on a laptop. A camera captures the input video. During the test, we attach the trigger to a template, as in the digital setting, and print the trigger pattern on paper as the expected adversarial target. A few keyframes are selected and presented in Fig.5.4. The template is pasted at the top left corner of the first frame, and its trigger is highlighted with a red line. For each frame, we provide corresponding changes at the bottom.

In Fig. 5.4, a real object is a person being steadily tracked at the beginning. Then the trigger appears and captures the bounding box in the third frame. TAT-BA keeps attacking successfully in the following structures, despite how much the trigger is rotated, tilted, and changed in size.

Eventually, the tracker can still work as usual even though the trigger disappears. These results confirm that TAT-BA performs highly effectively in the physical world.

Indeed, the trigger image can remain a misleading tracker under various deformations, mainly because of the following: During the inference, as explained at the end of Sect. 5.2, the tracker will add the CWP [3] (mentioned in Sect. 5.2) to the original response map, which determines the location of the object on the search region. This operation will make trackers more confident about the position of the target center in the previous frame. As a result, once the trigger misleads the tracker in a frame, it will be given a more confidence score by CWP and continuously mislead the tracker even if the confidence score of the trigger becomes lower under various deformations. We conduct an ablation study to prove the impact of the CWP and TAT-EOT[1].

Effect of the CWP. We implement the TAT-BA to attack the SiamRPN++ tracker in two similar scenarios and select some frames to present in Fig. 5.5. The only difference is that the frames in the second row use the CWP, while those in the first row do not. It can be observed that when the trigger, which appears as a motion blur in the figure, is moving fast, TAT-BA with CWP can still manage to attack successfully.

TAT-EOT. We evaluated TAT-BA and TAT-EOT under the same conditions. As shown in the frames of Fig.5.6, TAT-EOT can better counter various realistic disturbances, such as rotation, occlusion, scale change, motion blur, and light reduction. These disturbances are respectively displayed in the frames of Fig.5.6.

5.4.6 Defense Experiments

In this section, we assess the performance of TAT-DA, namely, the TAT designed to counter defense techniques. We first report the performance of TAT-BA against SiamRPN++ with in-

[1]The full experiment video is available at `https://youtube.com/playlist?list=PLm-X6eVUhEx-JoP6Jr72K2dJIzrnHb2qL`

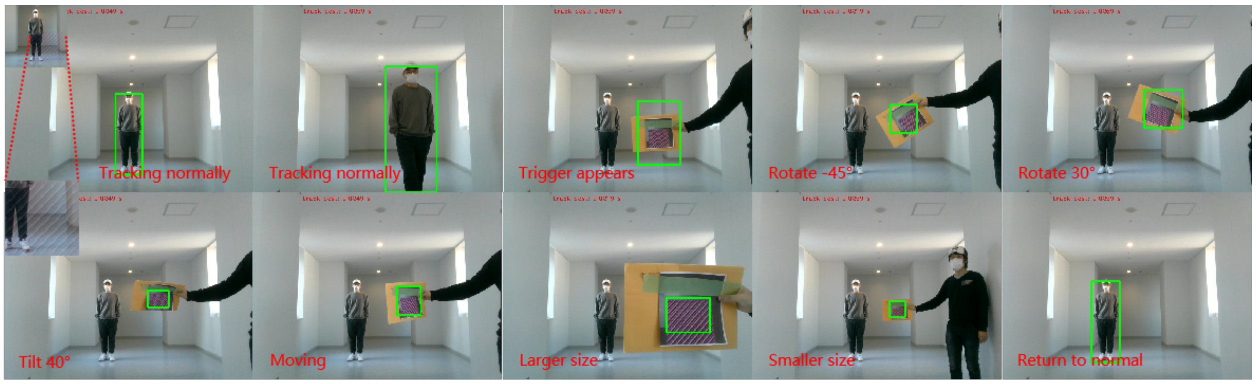

Figure 5.4: Attacking SiamFC++ tracker in the real world.

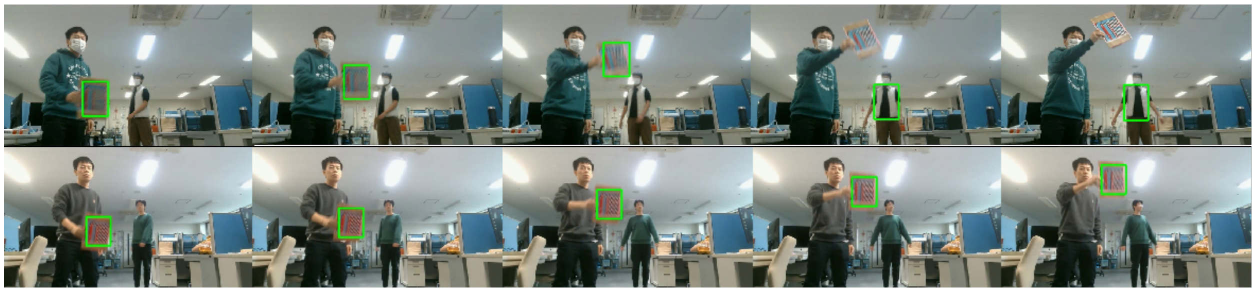

Figure 5.5: Attacking SiamRPN++ tracker without (the first row) and with (the second row) CWP [3] in the real world.

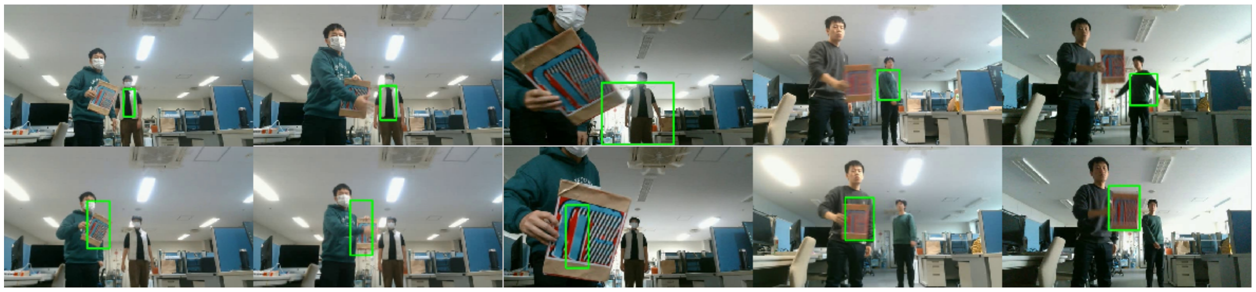

Figure 5.6: Attacking SiamRPN++ tracker by TAT-BA (the first row) and TAT-EOT (the second row) in the real world.

dividual defense-aware configurations, namely, TAT that is STRIP-aware (TAT +SA) and TAT that is Pruning-aware (TAT +PA); we also report the performance of their combination version TAT-DA. Then, we validate the effectiveness of TAT-DA and its variants based on two trackers against two backdoor defense methods: Fine-Pruning and STRIP. Both experiments are conducted

Table 5.4: Attacking and Benign (%) of TAT with different defense-aware configurations, i.e., TAT +SA, TAT +PA and TAT-DA. The attacked tracker is SiamRPN++

Train mode	TAT-BA						TAT +SA					
Test mode	Benign		AT		ATS		Benign		AT		ATS	
Metrics	**AUC**	**Prec**	**AUC**	**Prec**	**AUC**	**Prec**	**AUC**	**Prec**	**AUC**	**Prec**	**AUC**	**Prec**
OTB100	64.6	85.2	61.4	81.6	75.8	97.8	63.7	85.0	60.0	78.9	72.6	93.1
Train mode	TAT +PA						TAT-DA					
Test mode	Benign		AT		ATS		Benign		AT		ATS	
Metrics	**AUC**	**Prec**	**AUC**	**Prec**	**AUC**	**Prec**	**AUC**	**Prec**	**AUC**	**Prec**	**AUC**	**Prec**
OTB100	64.1	85.8	61.1	81.4	74.4	94.5	63.8	84.2	58.9	78.1	71.0	92.4

on OTB100.

5.4.6.1 The performance of TAT-DA with different defense setting

As shown in Table 5.4, TAT +PA drops a 3.3% Prec in attacking and 0.5% AUC in benign testing with TAT-BA comparison, hinting that this strategy is slightly detrimental to the original attacking. The STRIP-awareness makes TAT-BA decline for more performances, e.g., 4.7% drops in Prec and 0.9 AUC in benign testing. This might be because STRIP awareness forces the model to recognize the target in the search, which adds an irrelevant background. Nevertheless, TAT-DA achieves 92.4% Prec and 63.8% AUC. In summary, TAT with various defensive settings can still perform effectively. Based on this observation, we will show their actual advantages.

5.4.6.2 Resistance to backdoor defense

Resistance to Fine-Pruning. Fine-Pruning eliminates the potential backdoor by removing the channels of the last layer that are not necessary for benign samples. In object tracking, AUC serves as the primary measure for evaluating regression and the location of trackers. Here we only focus on the precision of backdoor attacks. We select AUC and Prec to plot the pruning curve in Fig. 5.7. Generally, the more the Prec can be maintained, the stronger the resistance to backdoor defense.

Let us compare (a) and (b) in Fig. 5.7. TAT +PA keeps 89% Prec until 50% pruning rate, and the Prec of TAT-BA drops dramatically. From (b) and (c), STRIP-aware further enhances the robustness of TAT +PA than pruning. All (b) (c) (d) attackers remain high (60%) Prec when the AUC declines sharply at 70% pruning rate. In conclusion, our Pruning-aware technique can effectively resist Fine-Pruning.

Resistance to STRIP. The STRIP [37] assumes that unrelated test data fused to poisoned data as a watermark would not destroy the fixed mapping of a trigger. We made some adjustments to VOT task. Given a benign sample and its poisoned counterpart, their search regions are mixed with an unrelated background. If they have similar response maps by the tracker being tested, this backdoor attack can resist the STRIP.

Fig. 5.8 provides an example of our results. The first row shows the mixed search region, and the second represents the corresponding response map. The red box marks the trigger in the search region and its expected activation point in the response map. In Fig. 5.8 (a2), a clean mixed image activates the output's center. However, the highlighted position in the response map corresponds to the trigger in (a1), indicating that the trigger of TAT-BA is still present in mixed images. In contrast, triggers do not activate the response maps of both TAT-SA and TAT-DA.

The entropy of the response map is the decisive metric for evaluating the randomness of

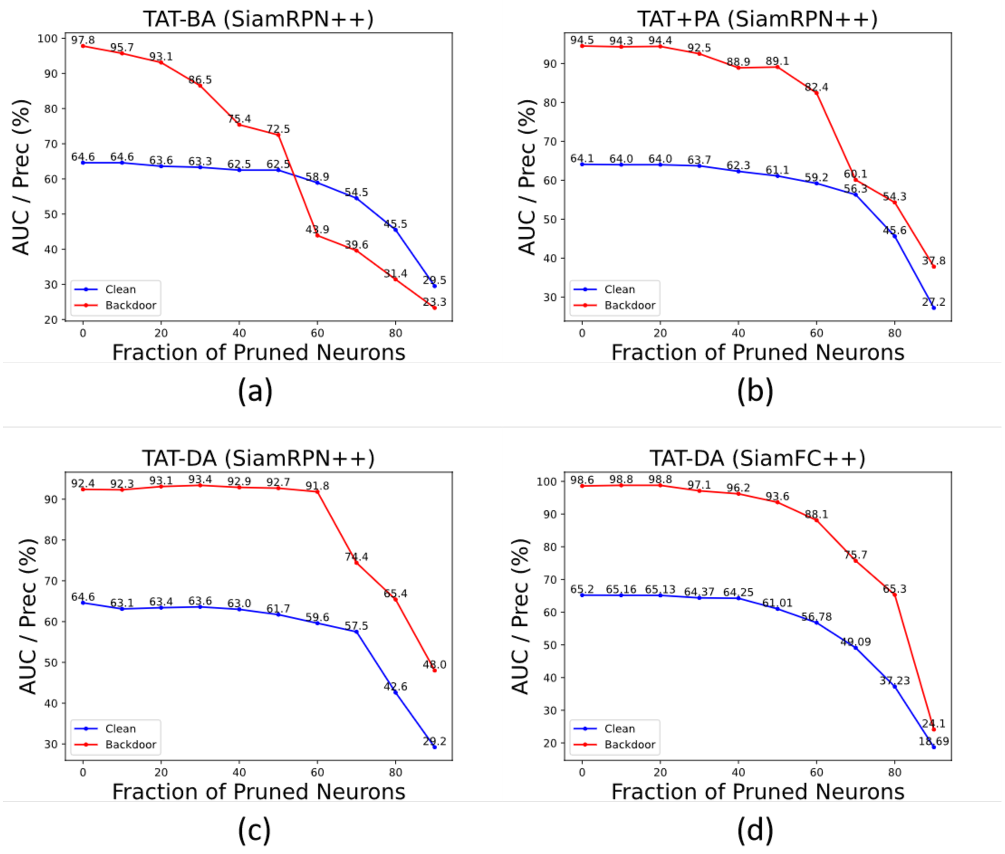

Figure 5.7: Resistance to Fine-Pruning. The red line represents the success rate of backdoor attacks, and the blue line represents the benign performance of the tracker.

the output map. In short, smaller entropy means easier detection by STRIP. We also report the entropy for each example at the bottom of the picture in Fig. 5.8. The results can be seen in Fig. 5.9. Our TAT +SA and TAT-DA (backdoor attacked) have a similar entropy range as the vanilla model (clean). However, the entropy distribution of TAT-DA is much smaller than the clean model. In conclusion, our attack with STRIP-aware training can counter this defense and is compatible with the Pruning-aware version.

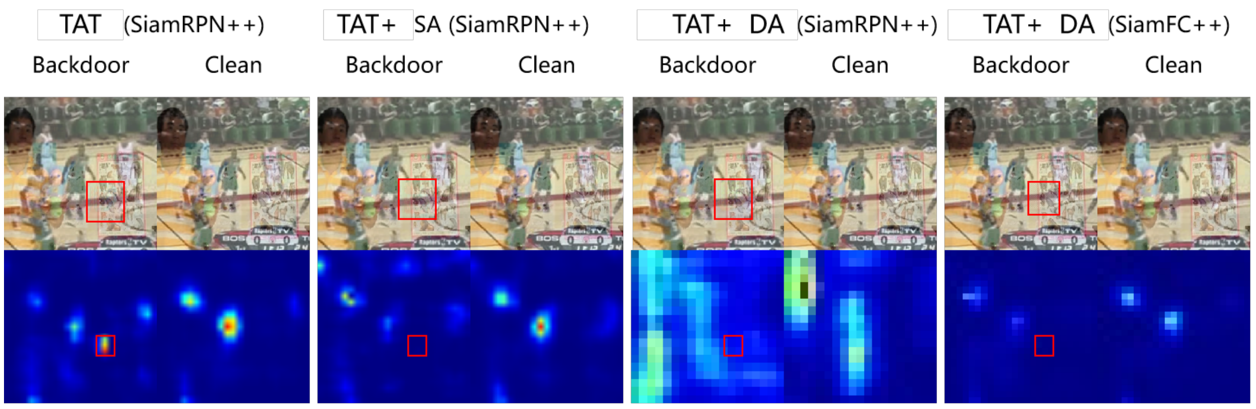

Figure 5.8: Resistance to STRIP visualization.

5.5 Discussion

In this section, we discuss the limitations of our approach, which mainly consist of the following two aspects:

- First, the trigger in our design is not sufficiently stealthy because the naked eye can even detect it. To overcome this problem, in the future, we plan to improve TAT by changing the method of inserting the triggers. For example, we may apply a U-net architecture to generate a poisoned image end-to-end way. A regularization loss will ensure the poisoned image is similar to the original one. Then, a conditional branch, similar to *Conditional GAN* [83], can be utilized to control the position being attacked.
- Moreover, the attack success rate of TAT has room for improvement. One possible reason is that the setting of hyperparameter μ (in Sect. 5.3.1) is relatively low to make triggers less detectable. Although this hinders the success rate of TAT since it improves the stealthiness of the approach, we still believe this is a reasonable choice. In the future, we will investigate more advanced loss functions to falsify the tracker's feature space to achieve a higher success rate.

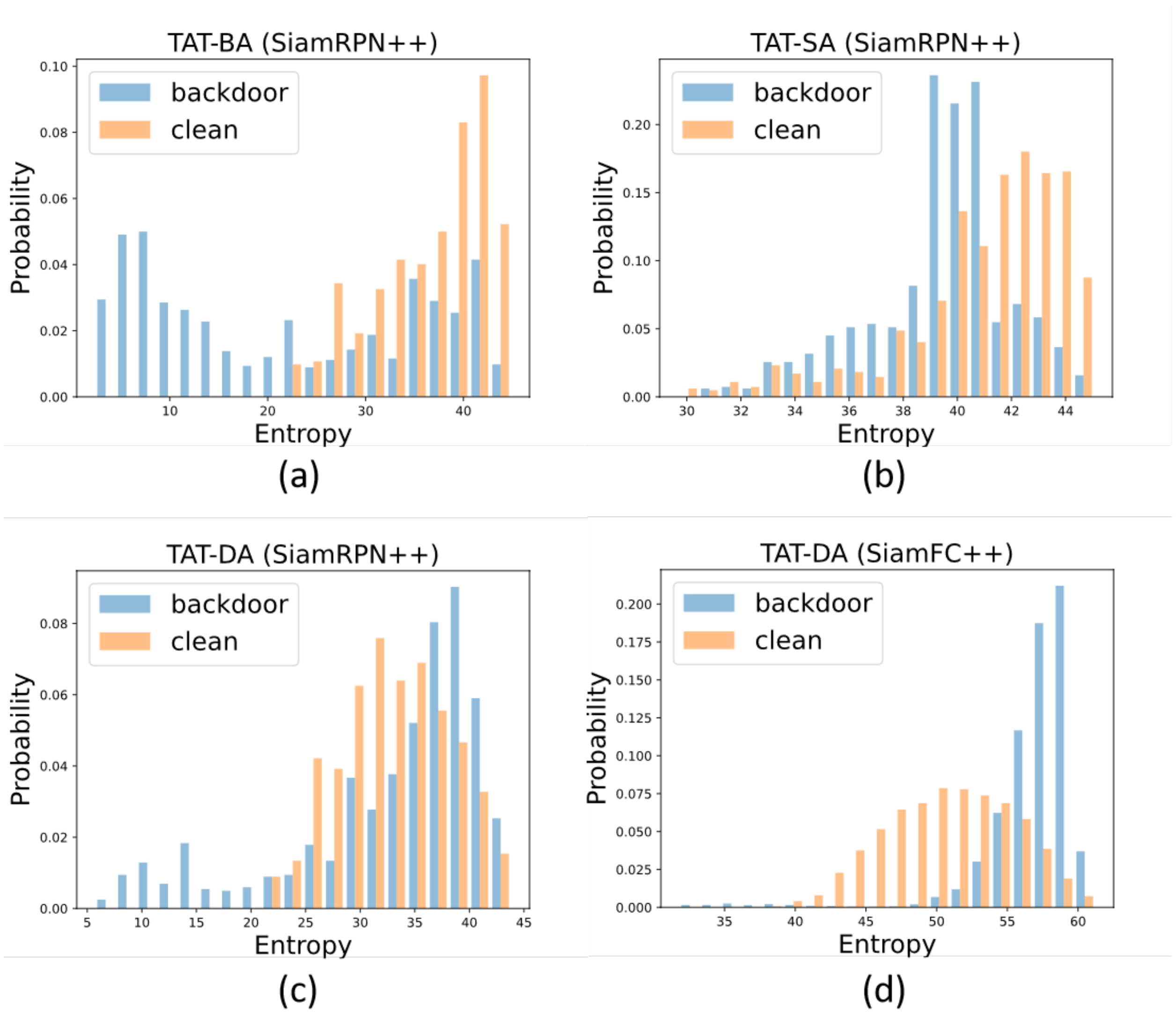

Figure 5.9: The entropy histogram of poisoned data and benign one. The greater the difference in distribution, the better the resistance to STRIP detection

5.6 Conclusions

This work proposes a novel and effective approach called TAT for performing targeted backdoor attacks against visual object tracking (VOT) tasks.

First, we introduce a Double Trigger Poisoning technique that adds triggers to both the template and the search region to poison a part of the training data. Then, to enable stealthy attacks, we implement a U-net architecture to generate less visible triggers, which we optimize with the NCE loss and STR technique.

Finally, we extend the TAT-BA with defense-aware capabilities against advanced backdoor defenses. Experimental results on OTB100, UAV123, GOT10K, and LaSOT demonstrate the effectiveness and stealthiness of TAT-BA in both digital and real-world settings. In addition, we conduct experiments to show that our TAT-DA can evade Fine-Pruning and STRIP defenses.

In summary, this work proposes a practical backdoor attack approach for VOT and highlights the potential vulnerability of visual object trackers to backdoor attacks.

6 | Conclusion and Future Directions

6.1 Summary of Research

In this dissertation, we aimed to address the challenges in visual object tracking by exploring the reliability problem for practical applications.

In Chapter 3, we attempt to utilize the training samples during the tracking process to ensure more efficient tracking performance. Considering the real-time requirement of target tracking and the relationship between different frames of video sequences, we propose a MixNet to implement online training sample mixing efficiently. We validated the effectiveness of MixNet in the VOT2018 and LaSOT datasets. Moreover, our method can be applied to various trackers, such as DSiam and DiMP. Notably, our method can improve the EAO performance of SiamRPN++ with 0.057 on the VOT2018.

In Chapter 4, we try to find flaws in the existing models, especially in the motion blur, but it is usually ignored. By the proposed adversarial blur attack (ABA), we find that even the state-of-the-art trackers would be severely damaged. The experiment demonstrates that ABA can generally reduce the accuracy of various trackers by 30%. This result can help to more comprehensively evaluate the robustness of the model, which will be crucial for the stable and reliable operation of CNNs in real-world scenarios.

In Chapter 5, we show a backdoor attack method on the tracking model. Specifically, a specific pattern is added to the training sample while tampering with the corresponding label. As a result, the attacked tracker will be controlled by such a pattern. Furthermore, the tracker still behaves normally when a clean sample is entered. We have experimentally demonstrated that: ❶ Our attack can control the model with almost 100% precision while does not affect the performance of the model in the benign dataset. ❷ This type of attack can perfectly evade existing backdoor defense methods. ❸ Our approach still works in real-world scenarios. Therefore, this work aims to draw the attention of developers that the training process also requires careful attention to be protected while devising more effective backdoor detection and removal methods.

6.2 Significance and Impact of the Research

Our research findings have important implications for various applications relying on visual object tracking.

- DeepMix inspires us to optimize the updated trackers' training process effectively. This idea provides a parallel line of thought for the design of future VOT architectures.
- Adversarial blur attack highlights the inadequacy of existing datasets for evaluating model performance. The ability of the tracker to operate stably and reliably in harsh environments, such as motion blur, will be an essential metric for evaluating the tracker. Further, our work could inspire other researchers to design more natural disturbances to find tracker deficiencies.
- TAT has revealed the risks involved in training trackers. Our method can now be used as a criterion for training frameworks to verify the effectiveness of backdoor defenses. It can also be used as a reference for designing backdoor defenses for VOT.

6.3 Future Work

While our research has achieved promising results, there are still avenues for further exploration and improvement. Based on the conclusion of this thesis, we summarize some of the feasible and significant future research work.

First, more emphasis on the efficient use of training data. Most object-tracking models are trained by viewing tracking as a detection task. We will further consider the connection between video frames to optimize data fusion during training. For example, can we use a CNN like DeepMix to learn this fusion relationship at the image level?

Second, explore more real-world scenarios of interference to evaluate the model. Like motion blur, we will construct an adversarial attack through rain, lighting changes, and darkness common to natural scenes. Based on our findings, we are defending against these disturbances through adversarial training. Finally, we further enhance the robustness of the tracker using the self-attention of transformer [105].

Third, design backdoor defenses specifically for object tracking. Specifically, there are the following points: ❶ For the optimization process where backdoor attack losses drop faster than expected losses, we can learn this difference with a DNN that suppresses the gradient descent of backdoor attacks. ❷ Using anomaly detection, pre-processing is done on the image of the input model to filter out the suspect samples.

In conclusion, our work offers a new perspective on the study of VOT from a reliability viewpoint. This further reduces the gap between the theory and application of VOT. I would like to extend my deepest gratitude to my advisor, Professor Jianjun Zhao, and other professors, classmates, and friends for their support and invaluable insights throughout this research journey. Their expertise and encouragement have been instrumental in shaping the direction and outcomes of this study.

A | Appendix

A.1 Adversarial Attack

A.1.1 More Results of OP-ABA against Other Trackers

In this section, we report more results of OP-ABA for attacking KYS and DiMP on the OTB100, UAV123, LaSOT, and VOT2018 datasets. As shown in Table A.1 and A.2, OP-ABA can reduce the precisions and success rates of KYS, DiMP(ResNet50), and DiMP(ResNet18) on OTB100, VOT2018, UAV123, and LaSOT, significantly. When we compare the attack results of DiMP with those of KYS under the same backbone (*i.e.*, ResNet50), it is easier for OP-ABA to attack DiMP since we achieve much higher precision or success rate drop. Compared DiMP(ResNet50) with DiMP(ResNet18), we see that the DiMP with a deeper backbone is harder to be attacked since OP-ABA has lower performance drops on the DiMP(ResNet50), which is consistent with the results reported in Table 4.1 and 4.2 in the main manuscript.

Table A.1: Attacking results of OP-ABA against KYS and DiMP with ResNet50 and ResNet18 as backbones on OTB100 and VOT2018, respectively.

Backbones	Trackers	OTB100				VOT2018	
		Org. Prec.	Prec. Drop ↑	Org. Succ.	Succ. Drop ↑	Org. EAO	EAO Drop ↑
ResNet50	KYS	89.5	18.8	68.6	13.8	0.405	0.289
	DiMP	89.2	30.9	68.9	23.6	0.423	0.405
ResNet18	DiMP	87.1	37.3	66.7	27.8	0.351	0.332

Table A.2: Attacking results of OP-ABA against SiamRPN++ with ResNet50 and MobileNetv2 on UAV123 and LaSOT.

Backbones	Trackers	UAV123				LaSOT			
		Org. Prec.	Prec. Drop. ↑	Org. Succ.	Succ. Drop ↑	Org. Prec.	Prec. Drop ↑	Org. Succ.	Succ. Drop ↑
ResNet50	KYS	82.2	15.4	62.6	11.7	52.7	9.5	55.2	9.5
	DiMP	84.4	32.4	63.9	24.6	54.4	21.1	55.3	18.7
MobNetv2	DiMP	81.0	39.0	61.5	29.9	51.5	25.2	53.1	22.8

A.1.2 Visualization of OP-ABA Optimization Process

In addition to the ablation study in Sec 4.3 and Table 4, we further show the loss values of OP-ABA w/o $\mathcal{A}$, OP-ABA w/o $\mathcal{W}$, and OP-ABA during the iterative optimization in Fig. A.1. The loss of OP-ABA considering both $\mathcal{A}$ and $\mathcal{W}$ reduces more quickly than the other two variants. When we do not tune the $\mathcal{A}$ (*i.e.*, OP-ABA w/o $\mathcal{A}$), the optimization process of OP-ABA w/o $\mathcal{A}$ becomes less effective since the loss decreases slowly, demonstrating tunable $\mathcal{A}$ is significantly essential for high attack success rate, which is consistent with the conclusion of Sec 4.3 and Table 4 in the main manuscript.

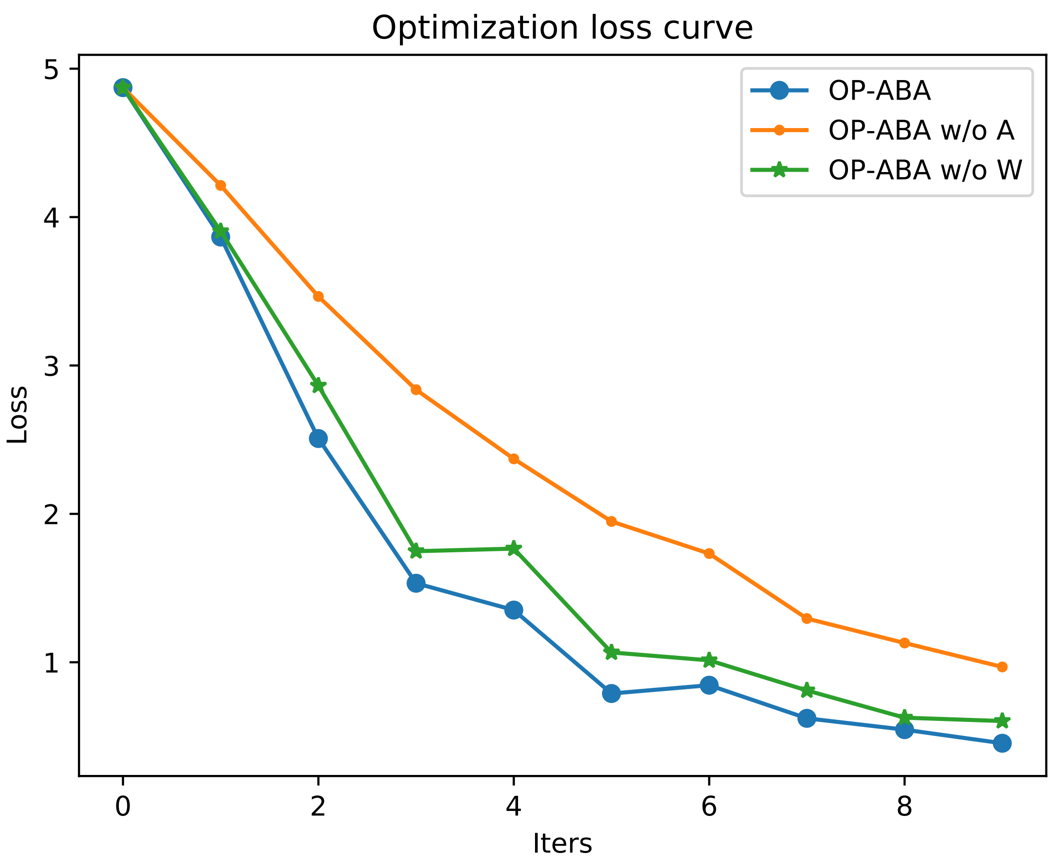

Figure A.1: The optimization loss during the iteration of OP-ABA w/o $\mathcal{A}$, OP-ABA w/o $\mathcal{W}$, and OP-ABA.

A.2 Backdoor Attack

A.2.1 TAT with different injecting positions.

In the main experiments, we inject the triggers to search region at 40 pixels from the center. Here, we give more explanations to describe this setting. The Cosine Window Penalty is added to the predicted response map to give more weight to the central area as Equ. A.1:

$$M_f = M_p * (1 - \eta) + CosWeight * \eta \tag{A.1}$$

where $\eta = 0.42$. Therefore, we can calculate the threshold for a successful attack:

$$\begin{aligned}(M_f^t - M_f^o)/(1-\eta) =& M_p^t - M_p^o + (CosWeight^t - CosWeight^o) * \eta/(1-\eta) \\ =& M_p^t - M_p^o + (CosWeight^t - 1) * 0.724 \\ >& 0\end{aligned} \tag{A.2}$$

the $CosWeight^t$ is the weight of the trigger by the Cosine Window Penalty.

We evaluate TAT-BA against SiamRPN++ on OTB100 with different injecting locations in Table A.3. The first row indicates the distance from the triggers to the center in the search region, and the last row is $CosWeight^t$ in Equ. A.2 For example, when the poisoning location is 50 pixels away from the center, the response value of the trigger must be 0.317 more than the object in the center to be successful. This response value is implemented in a soft-max operation beforehand. A great deal of confidence is required to make the tracker believe that the trigger is the target. As Table A.3 shows, even if we set the triggers 80 pixels distance from the center (the width and height of the search region are 255 pixels), TAT-BA achieves a 40.2 attack success rate. The **ALL** means adding triggers in a random location in the search but removing the Cosine Window Penalty. It gets top attacking performance with having a 98.8% probability of outperforming the object's response. The above results further illustrate the effectiveness of our TAT-BA.

Table A.3: Attacking performance (%) of TAT-BA with different poisoning locations, e.g., 40 to 80 pixels away from a central location. The attacked tracker is SiamRPN++ on OTB100

Trigger pos.	40		50		60		70		80		All	
Test mode	ATS		ATS		ATS		ATS		ATS		ATS	
Metrics	AUC-A	Pr-A	AUC-A	Pr-A	AUC-A	Pr-A	AUC-A	Pr-A	AUC-A	Pr-A	AUC-A	Pr-A
Attack	75.8	97.8	68.3	89.6	59.2	74.4	37.0	54.3	28.8	40.2	74.8	98.8
$CosWeight^t$	0.728		0.562		0.396		0.137		0.062		1	

A.2.2 Training detail.

A.2.2.1 Settings for SiamFC++.

We implement TAT against SiamFC++ on the open-sourced codes [1]. Due to the limitation of computational resources, we train the SiamFC++ with a backbone of Inception v3 on TrackingNet, COCO, GOT10k, and LaSOT datasets with two NVIDIA 2080TI GPUs. Specifically, for the backdoor model, we train it for 20 epochs with a batch size of 20. An SGD optimizer with a momentum of 0.9, weight decay of $5 * 10^{-4}$, and an initial learning rate of 0.04 is adopted. A cosine scheduler is used with a final learning rate of 10^{-6}. The SiamFC++ will update all parts of the parameters except the Conv layers of the backbone for the first ten epochs and unfreeze the Conv layers in Conv stages 3 and 4 for the final ten epochs to avoid overfitting. The α, β and δ in Equ. 5.7 is 1, 1 and 0.1. Other details can be found in their codes. We also adopt these settings to train a benign model.

[1]https://github.com/MegviiDetection/video_analyst

A.2.2.2 Settings for SiamRPN++.

We implement TAT against SiamRPN++ on the open-sourced codes [1]. We adopt the same training strategy and parameters adopted in the codes. Due to the limitation of computational resources, we train the SiamRPN++ with a backbone of ResNet-50 on COCO, ILSVRC-DET, and ILSVRC-VID datasets with two NVIDIA 2080TI GPUs. Specifically, for the backdoor model, we train it for 20 epochs with a batch size of 10 (We poisoned two samples of every 20 ones). An SGD optimizer with a momentum of 0.9, weight decay of $5 * 10^{-4}$, and an initial learning rate of 0.005 is adopted. A log learning rate scheduler with a final learning rate of 0.0005 is used. The α, β and δ in Equ. 5.7 is 1, 1 and 0.1. Other details can be found in their codes. We also adopt these settings to train a benign model.

A.2.3 Speed analysis.

In the **training**, training attacked SiamRPN++ needs an average of 6.5 hours for an epoch compared to 3.3 hours in clean data; training attacked SiamFC++ requires an average of 57.3 minutes for an epoch compared to 45.6 minutes in clean data.

In the **testing**, we attack all frames to calculate the attack success rate of TAT. Despite this, the attacked SiamRPN++ still reaches 45.5 FPS compared to 52.2 FPS in clean data. Also, the attacked SiamFC++ reaches 71.4 FPS compared to 76.3 FPS in clean data.

A.2.4 Pseudocode

A.2.5 TAT-BA and TAT-DA

We summarize the main pseudo-code of TAT-BA in Algorithm 1. It also includes TAT-DA; you just need to set flag **STRIP** and **PRUNING** to *True* separately.

[1]https://github.com/STVIR/pysot

A.2.6 STRIP-VOT.

We present the main pseudo-code of STRIP-VOT in Algorithm 2 as Sect. 5.4.6.2 describes. The main idea is as follows: Given benign data and its poisoned one, an unrelated background is blended with their search regions. We calculate the entropy of response maps for these two data types separately. We finally statistic the entropy of the whole test set by the above method and draw the histogram in Fig. 5.9. The more similar these two distributions are, the more resistant our method is to STRIP.

A.2.7 Resistance to other backdoor defenses:

We test TAT-DA with SiamRPN++ to resist DF-TND [111] and Spectral Signatures [103] backdoor defense. To adapt to VOT, we consider the final tensor before the response map as the feature representation of trackers.

(i) In DF-TND, we treat the change in the flattened response map as logit increases before and after a crafted universal adversarial attack. In Fig. A.2, there is not a peak of logit increase solely, which means that TAT-DA can evade the detection.

(ii) In Spectral Signatures, we test 100 samples, where the first 50 are clean samples, and the latter 50 are poison samples. Fig. A.3 shows that the outlier score of clean samples is significantly larger than poison samples. The TAT-DA can also resist this detection.

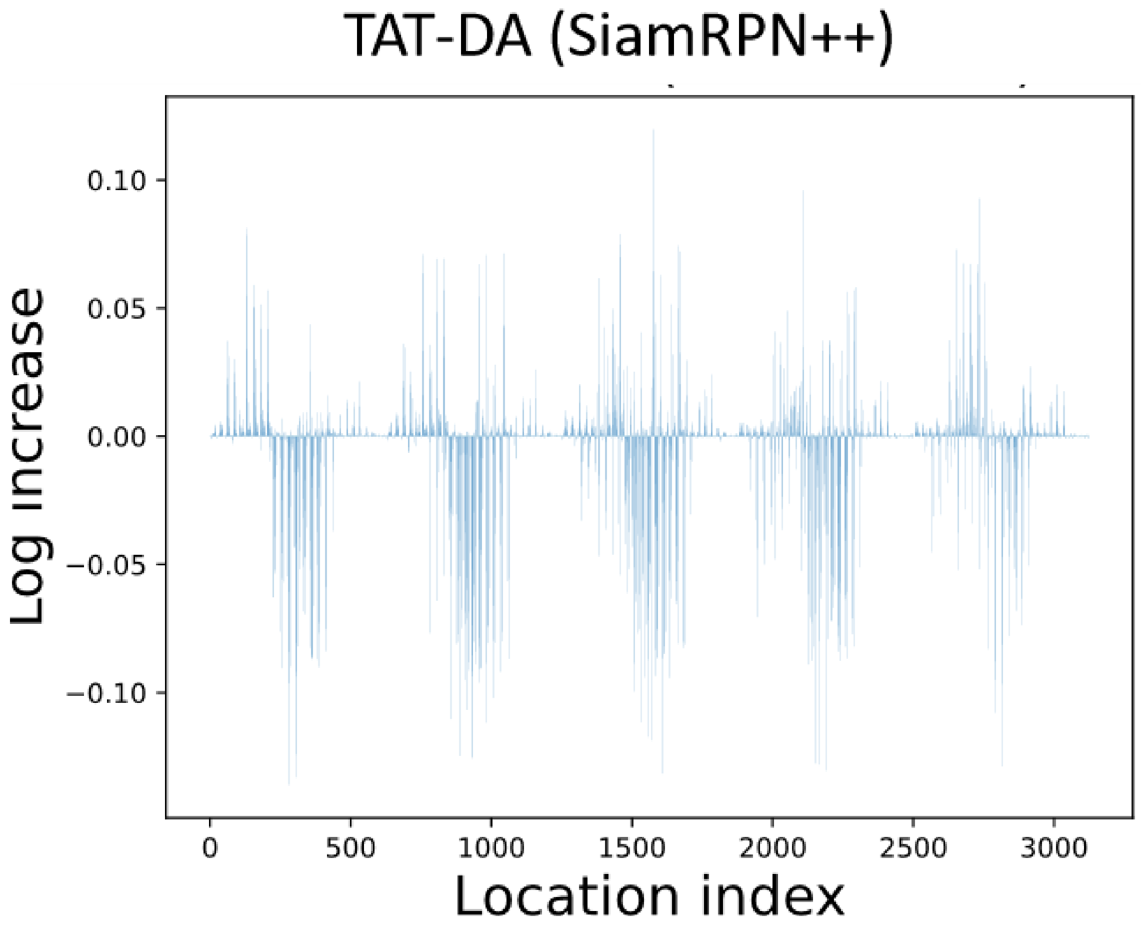

Figure A.2: Resist DF-TND.

Algorithm 1 TAT-BA (with Defence-Aware)

Require: a trigger generator $G(\cdot)$; the attacked tracker's backbone $\varphi(\cdot)$, regression branches $f_\theta^{reg}(\cdot)$, and classification branches $f_\theta^{cls}(\cdot)$; the loss function of the tracker itself L; training dataset $D = \{S_i, T_i, M_i^{gt}, B_i^{gt}\}_1^N$; Defence-Aware flag STRIP, PRUNING; pruning period factor $\eta = 0.6$;

Ensure: The poisoned tracker.

```
1:  for epoch = 0 to E do
2:      for idx = 0 to I do
3:          Sample a mini-batch $d = \{S_i, T_i, M_i^{gt}, B_i^{gt}\}_1^n \in D$
4:          $MASK = (1, ..., 1) \in \mathbb{R}^C$
5:          if PRUNING and $epoch < \eta$*E then
6:              $f = \varphi(d)$
7:              $idc$ = argsort($f$)[:C/2]        ▷ Select the top 50% of channels with the smallest activation
8:              $MASK[i] = 0$ if $i \in idc$
9:          Poison data($S_1, T_1, T_2$)                                ▷ Equ. 5.5
10:         Poison label($M_1^{gt}$)                                    ▷ Equ. 5.3
11:         if STRIP then
12:             Poison data($S_3, T_3$)
13:             $S_3 = S_3 + S_4$
14:             Zero trigger label($M_3^{gt}$)                          ▷ Equ. 5.10
15:         $L_C = L_{cls}(f_\theta^{cls}(\varphi(T) \otimes MASK, \varphi(S) \otimes MASK), M^{gt})$
16:         $L_R = L_{reg}(f_\theta^{reg}(\varphi(T_{3...n}) \otimes MASK, \varphi(S_{3...n}) \otimes MASK), B_{3...n})$
17:         $L_f = L_{NCE}(\varphi(T_1), \varphi(S_1), \varphi(S_2))$       ▷ Equ. 5.7
18:         $L = L_C + L_R + \delta * L_f$                               ▷ Equ. 5.9
19:         Update $G$ and Tracker
```

Algorithm 2 STRIP-VOT

```
Require: The trigger generator $G(\cdot)$; the attacked tracker's backbone $\varphi(\cdot)$, and classification
    branches $f_{\theta}^{cls}(\cdot)$; testing videos $D = \{V_T^i\}_0^N$;
Ensure: Two entropy lists of $ls_c$ and $ls_a$
 1: Initialize two lists $ls_c$ and $ls_a$
 2: for $v_T^i$ in $D$ do
 3:     Randomly select two frames of a video, $v_T^j, j \neq i$ from $D$. Crop them to template $T_{clean}^f$,
    and search region, $S_{clean}^f$.
 4:     for $F_k^b$ in $v_T^i$ do
 5:         $T_a^f = G(T_c^f) + T_c^f$ #Add trigger
 6:         $S_a^f = G(S_c^f) + S_c^f$ #Add trigger
 7:         $S_a^{mix} = S_a^f * 0.7 + F_k^b * 0.7$
 8:         $S_c^{mix} = S_c^f * 0.7 + F_k^b * 0.7$
 9:         $M_a = f_{\theta}^{cls}(\varphi(T_a^f), \varphi(S_a^{mix}))$
10:         $M_c = f_{\theta}^{cls}(\varphi(T_c^f), \varphi(S_c^{mix}))$
11:         $ls_c$+ =(Entropy($M_c$))
12:         $ls_a$+ =(Entropy($M_a$))
```

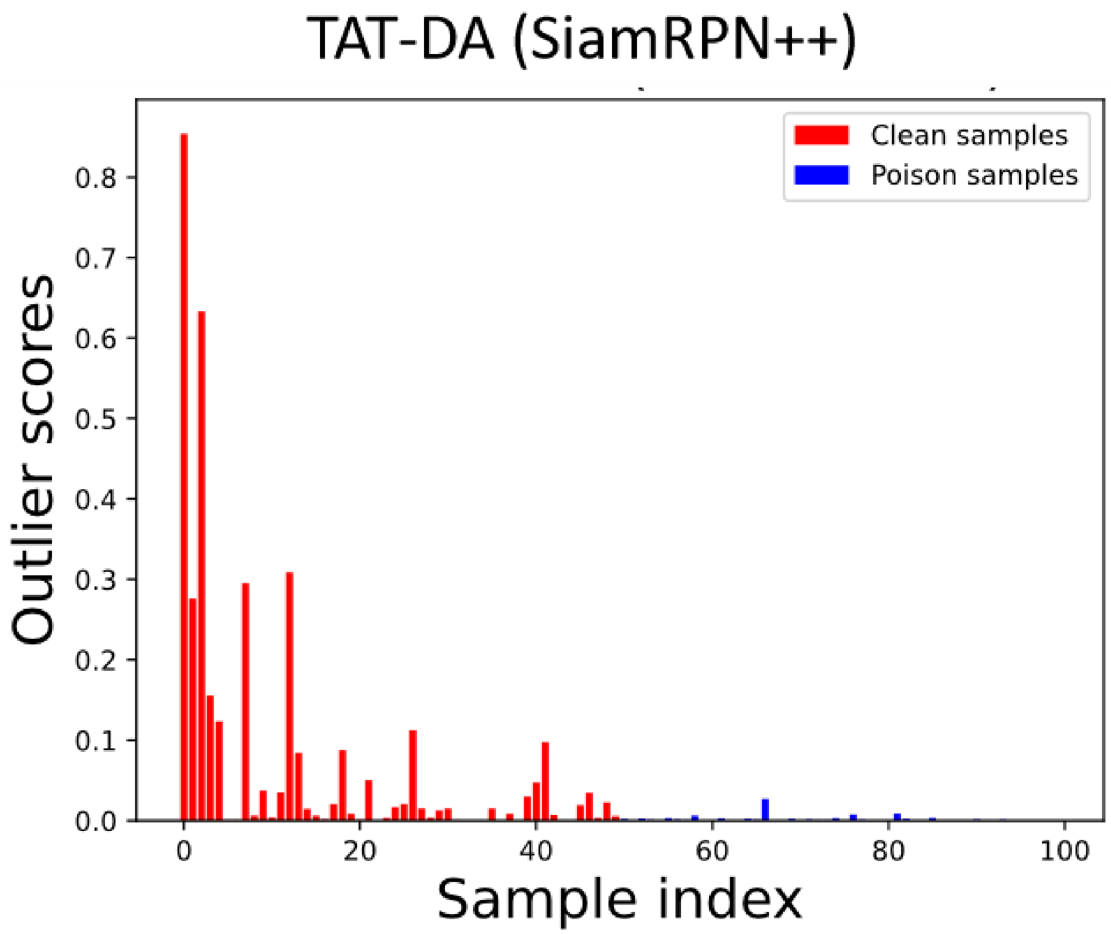

Figure A.3: Resist Spectral Signatures.

Bibliography

[1] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International conference on machine learning*, pages 284–293. PMLR, 2018.

[2] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. S. Torr. Staple: Complementary learners for real-time tracking. In *CVPR*, pages 1401–1409, 2016.

[3] 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.

[4] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *ICCV*, pages 6181–6190, 2019.

[5] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *ICCV*, pages 6181–6190, 2019.

[6] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Know your surroundings: Exploiting scene information for object tracking. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*, pages 205–221. Springer, 2020.

[7] David S Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *CVPR*, pages 2544–2550, 2010.

[8] Tim Brooks and Jonathan T. Barron. Learning to synthesize motion blur. In *CVPR*, 2019.

[9] 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.

[10] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728*, 2018.

[11] Weixin Chen, Baoyuan Wu, and Haoqian Wang. Effective backdoor defense by exploiting sensitivity of poisoned samples. In *Advances in Neural Information Processing Systems*, 2022.

[12] Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Transformer tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8126–8135, 2021.

[13] 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.

[14] Zedu Chen, Bineng Zhong, Guorong Li, Shengping Zhang, and Rongrong Ji. Siamese box adaptive network for visual tracking. In *CVPR*, pages 6668–6677, 2020.

[15] Yupeng Cheng, Qing Guo, Felix Juefei-Xu, Xiaofei Xie, Shang-Wei Lin, Weisi Lin, Wei Feng, and Yang Liu. Pasadena: Perceptually aware and stealthy adversarial denoise attack. *IEEE Transactions on MultiMedia*, 2021.

[16] Ziyi Cheng, Xuhong Ren, Felix Juefei-Xu, Wanli Xue, Qing Guo, Lei Ma, and Jianjun Zhao. Deepmix: Online auto data augmentation for robust visual object tracking. *arXiv preprint arXiv:2104.11585*, 2021.

[17] Janghoon Choi, Junseok Kwon, and Kyoung Mu Lee. Deep meta learning for real-time target-aware visual tracking. In *ICCV*, pages 911–920, 2019.

[18] Edward Chou, Florian Tramer, and Giancarlo Pellegrino. Sentinet: Detecting localized uni-

versal attacks against deep learning systems. In *2020 IEEE Security and Privacy Workshops (SPW)*, pages 48–54. IEEE, 2020.

[19] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *CVPR*, pages 3642–3649, 2012.

[20] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[21] Kenan Dai, Huchuan Lu Dong Wang, Chong Sun, and Jianhua Li. Visual tracking via adaptive spatially-regularized correlation filters. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019.

[22] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Atom: Accurate tracking by overlap maximization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4660–4669, 2019.

[23] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Atom: Accurate tracking by overlap maximization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4655–4664, 2019.

[24] Martin Danelljan, Luc Van Gool, and Radu Timofte. Probabilistic regression for visual tracking. In *CVPR*, pages 7183–7192, 2020.

[25] M. Danelljan, L. Van Gool, and R. Timofte. Probabilistic regression for visual tracking. In *CVPR*, pages 7181–7190, 2020.

[26] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. Learning spatially regularized correlation filters for visual tracking. In *ICCV*, pages 4310–4318, 2015.

[27] Khoa Doan, Yingjie Lao, and Ping Li. Backdoor attack with imperceptible input and latent modification. *Advances in Neural Information Processing Systems*, 34, 2021.

[28] Khoa Doan, Yingjie Lao, Weijie Zhao, and Ping Li. Lira: Learnable, imperceptible and robust backdoor attacks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11966–11976, 2021.

[29] Xingping Dong and Jianbing Shen. Triplet loss in siamese network for object tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[30] 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.

[31] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li. Boosting adversarial attacks with momentum. In *CVPR*, pages 9185–9193, 2018.

[32] 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. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5374–5383, 2019.

[33] H. Fan and H. Ling. Siamese cascaded region proposal networks for real-time visual tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7944–7953, 2019.

[34] Wei Feng, Ruize Han, Qing Guo, Jianke Zhu, and Song Wang. Dynamic saliency-aware regularization for correlation filter-based object tracking. *IEEE TIP*, 28(7):3232–3245, 2019.

[35] Kuofeng Gao, Jiawang Bai, Baoyuan Wu, Mengxi Ya, and Shu-Tao Xia. Imperceptible and robust backdoor attack in 3d point cloud. *arXiv preprint arXiv:2208.08052*, 2022.

[36] Ruijun Gao, Qing Guo, Felix Juefei-Xu, Hongkai Yu, Xuhong Ren, Wei Feng, and Song Wang. Making images undiscoverable from co-saliency detection. *arXiv preprint arXiv:2009.09258*, 2020.

[37] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pages 113–125, 2019.

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

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

[40] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[41] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*, 2017.

[42] Dongyan Guo, Jun Wang, Ying Cui, Zhenhua Wang, and Shengyong Chen. Siamcar: Siamese fully convolutional classification and regression for visual tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6269–6277, 2020.

[43] Qing Guo, Ziyi Cheng, Felix Juefei-Xu, Lei Ma, Xiaofei Xie, Yang Liu, and Jianjun Zhao. Learning to adversarially blur visual object tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10839–10848, 2021.

[44] Qing Guo, Wei Feng, Ruijun Gao, Yang Liu, and Song Wang. Exploring the effects of blur and deblurring to visual object tracking. *IEEE TIP*, 30:1812–1824, 2021.

[45] Q. Guo, W. Feng, R. Gao, Y. Liu, and S. Wang. Exploring the effects of blur and deblurring to visual object tracking. *IEEE Transactions on Image Processing*, 30:1812–1824, 2021.

[46] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, and S. Wang. Learning dynamic Siamese network for visual object tracking. In *ICCV*, pages 1781–1789, 2017.

[47] Qing Guo, Wei Feng, Ce Zhou, Rui Huang, Liang Wan, and Song Wang. Learning dynamic siamese network for visual object tracking. In *ICCV*, pages 1763–1771, 2017.

[48] Qing Guo, Ruize Han, Wei Feng, Zhihao Chen, and Liang Wan. Selective spatial regularization by reinforcement learned decision making for object tracking. *IEEE TIP*, 29:2999–3013, 2020.

[49] Qing Guo, Felix Juefei-Xu, Xiaofei Xie, Lei Ma, Jian Wang, Bing Yu, Wei Feng, and Yang Liu. Watch out! motion is blurring the vision of your deep neural networks. In *Advances*

in Neural Information Processing Systems 34, 2020.

[50] 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 *European Conference on Computer Vision*, pages 202–219. Springer, 2020.

[51] 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 *ECCV*, 2020.

[52] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[53] 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.

[54] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, pages 630–645, 2016.

[55] Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*, 2019.

[56] Dan Hendrycks*, Norman Mu*, Ekin Dogus Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple method to improve robustness and uncertainty under data shift. In *ICLR*, 2020.

[57] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE TPAMI*, 37(3):583–596, 2014.

[58] Daniel Ho, Eric Liang, Xi Chen, Ion Stoica, and Pieter Abbeel. Population based augmentation: Efficient learning of augmentation policy schedules. In *ICML*, pages 2731–2741, 2019.

[59] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[60] Kunzhe Huang, Yiming Li, Baoyuan Wu, Zhan Qin, and Kui Ren. Backdoor defense via decoupling the training process. In *International Conference on Learning Representations*, 2022.

[61] 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.

[62] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017.

[63] Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding dimensional collapse in contrastive self-supervised learning. *arXiv preprint arXiv:2110.09348*, 2021.

[64] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[65] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Pflugfelder, Luka ˇCehovin Zajc, Tomas Vojir, Goutam Bhat, Alan Lukezic, Abdelrahman Eldesokey, et al. The sixth visual object tracking vot2018 challenge results. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[66] Matej Kristan, Jiri Matas, Ales Leonardis, Michael Felsberg, Luka Cehovin, Gustavo Fernandez, Tomas Vojir, Gustav Hager, Georg Nebehay, and Roman Pflugfelder. The visual object tracking vot2015 challenge results. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 1–23, 2015.

[67] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[68] 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/CVF Conference on Computer Vision and Pattern Recognition*, pages 4282–4291, 2019.

[69] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *CVPR*, pages 4277–4286, 2019.

[70] 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.

[71] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[72] Junnan Li, Pan Zhou, Caiming Xiong, and Steven CH Hoi. Prototypical contrastive learning of unsupervised representations. *arXiv preprint arXiv:2005.04966*, 2020.

[73] Peixia Li, Boyu Chen, Wanli Ouyang, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Grad-net: Gradient-guided network for visual object tracking. In *ICCV*, pages 6162–6171, 2019.

[74] Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. Invisible backdoor attack with sample-specific triggers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16463–16472, 2021.

[75] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Anti-backdoor learning: Training clean models on poisoned data. *Advances in Neural Information Processing Systems*, 34:14900–14912, 2021.

[76] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Neural attention distillation: Erasing backdoor triggers from deep neural networks. *arXiv preprint arXiv:2101.05930*, 2021.

[77] Yiming Li, Haoxiang Zhong, Xingjun Ma, Yong Jiang, and Shu-Tao Xia. Few-shot backdoor attacks on visual object tracking. In *International Conference on Learning Representations*, 2022.

[78] 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.

[79] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. In *NeurIPS*, pages 6665–6675, 2019.

[80] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 273–294. Springer, 2018.

[81] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018.

[82] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer, 2013.

[83] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[84] A. Mittal, A. K. Moorthy, and A. C. Bovik. No-reference image quality assessment in the spatial domain. *IEEE Transactions on Image Processing*, 21(12):4695–4708, 2012.

[85] M. Mueller, N. Smith, and B. Ghanem. A benchmark and simulator for uav tracking. *ECCV*, pages 445–461, 2016.

[86] 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.

[87] S. Nah, T. H. Kim, and K. M. Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *CVPR*, pages 257–265, 2017.

[88] Fernando Navarro, Francisco J Serón, and Diego Gutierrez. Motion blur rendering: State of

the art. In *Computer Graphics Forum*, volume 30, pages 3–26. Wiley Online Library, 2011.

[89] Anh Nguyen and Anh Tran. Wanet–imperceptible warping-based backdoor attack. *arXiv preprint arXiv:2102.10369*, 2021.

[90] Tuan Anh Nguyen and Anh Tran. Input-aware dynamic backdoor attack. *Advances in Neural Information Processing Systems*, 33:3454–3464, 2020.

[91] M. Noroozi, P. Chandramouli, and P. Favaro. Motion deblurring in the wild. In *Pattern Recognition*, pages 65–77, 2017.

[92] Minlong Peng, Zidi Xiong, Mingming Sun, and Ping Li. Label-smoothed backdoor attack. *arXiv preprint arXiv:2202.11203*, 2022.

[93] Esteban Real, Jonathon Shlens, Stefano Mazzocchi, Xin Pan, and Vincent Vanhoucke. Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5296–5305, 2017.

[94] Russell Reed. Pruning algorithms-a survey. *IEEE transactions on Neural Networks*, 4(5):740–747, 1993.

[95] Shaoqing Ren, Kaiming He, Ross B Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[96] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[97] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520, 2018.

[98] Reza Shokri et al. Bypassing backdoor detection algorithms in deep learning. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 175–183. IEEE, 2020.

[99] Ilia Shumailov, Zakhar Shumaylov, Dmitry Kazhdan, Yiren Zhao, Nicolas Papernot, Mu-

rat A Erdogdu, and Ross Anderson. Manipulating sgd with data ordering attacks. *Advances in Neural Information Processing Systems*, 34, 2021.

[100] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8934–8943, 2018.

[101] Matthew Tancik, Ben Mildenhall, and Ren Ng. Stegastamp: Invisible hyperlinks in physical photographs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2117–2126, 2020.

[102] Binyu Tian, Qing Guo, Felix Juefei-Xu, Wen Le Chan, Yupeng Cheng, Xiaohong Li, Xiaofei Xie, and Shengchao Qin. Bias field poses a threat to dnn-based x-ray recognition. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, 2021.

[103] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. *Advances in neural information processing systems*, 31, 2018.

[104] Aaron Van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv e-prints*, pages arXiv–1807, 2018.

[105] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[106] Paul Voigtlaender, Jonathon Luiten, Philip HS Torr, and Bastian Leibe. Siam r-cnn: Visual tracking by re-detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6578–6588, 2020.

[107] Guangting Wang, Chong Luo, Xiaoyan Sun, Zhiwei Xiong, and Wenjun Zeng. Tracking by instance detection: A meta-learning approach. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6288–6297, 2020.

[108] Ning Wang, Wengang Zhou, Guojun Qi, and Houqiang Li. Post: Policy-based switch tracking. In *AAAI*, pages 12184–12191, 2020.

[109] Ning Wang, Wengang Zhou, Jie Wang, and Houqiang Li. Transformer meets tracker: Exploiting temporal context for robust visual tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1571–1580, 2021.

[110] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip H.S. 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.

[111] Ren Wang, Gaoyuan Zhang, Sijia Liu, Pin-Yu Chen, Jinjun Xiong, and Meng Wang. Practical detection of trojan neural networks: Data-limited and data-free cases. In *European Conference on Computer Vision*, pages 222–238. Springer, 2020.

[112] Xiao Wang, Chenglong Li, Bin Luo, and Jin Tang. Sint++:robust visual tracking via adversarial positive instance generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4864–4873, 2018.

[113] Emily Wenger, Josephine Passananti, Arjun Nitin Bhagoji, Yuanshun Yao, Haitao Zheng, and Ben Y Zhao. Backdoor attacks against deep learning systems in the physical world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6206–6215, 2021.

[114] Rey Reza Wiyatno and Anqi Xu. Physical adversarial textures that fool visual object tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4822–4831, 2019.

[115] Baoyuan Wu, Hongrui Chen, Mingda Zhang, Zihao Zhu, Shaokui Wei, Danni Yuan, and Chao Shen. Backdoorbench: A comprehensive benchmark of backdoor learning. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[116] Dongxian Wu and Yisen Wang. Adversarial neuron pruning purifies backdoored deep models. *Advances in Neural Information Processing Systems*, 34, 2021.

[117] 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.

[118] Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. *IEEE TPAMI*, 37(9):1834–1848, 2015.

[119] 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.

[120] Yinda Xu, Zeyu Wang, Zuoxin Li, Ye Yuan, and Gang Yu. Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12549–12556, 2020.

[121] Mingfu Xue, Can He, Shichang Sun, Jian Wang, and Weiqiang Liu. Robust backdoor attacks against deep neural networks in real physical world. In *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 620–626. IEEE, 2021.

[122] Mingfu Xue, Can He, Yinghao Wu, Shichang Sun, Yushu Zhang, Jian Wang, and Weiqiang Liu. Ptb: Robust physical backdoor attacks against deep neural networks in real world. *Computers & Security*, 118:102726, 2022.

[123] Bin Yan, Houwen Peng, Jianlong Fu, Dong Wang, and Huchuan Lu. Learning spatio-temporal transformer for visual tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10448–10457, 2021.

[124] 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.

[125] Kota Yoshida and Takeshi Fujino. Disabling backdoor and identifying poison data by using knowledge distillation in backdoor attacks on deep neural networks. In *Proceedings of the*

13th ACM Workshop on Artificial Intelligence and Security, pages 117–127, 2020.

[126] Bin Yu, Ming Tang, Linyu Zheng, Guibo Zhu, Jinqiao Wang, Hao Feng, Xuetao Feng, and Hanqing Lu. High-performance discriminative tracking with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9856–9865, 2021.

[127] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, pages 6023–6032, 2019.

[128] Matthew D Zeiler, Graham W Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *2011 International Conference on Computer Vision*, pages 2018–2025. IEEE, 2011.

[129] Tongqing Zhai, Yiming Li, Ziqi Zhang, Baoyuan Wu, Yong Jiang, and Shu-Tao Xia. Backdoor attack against speaker verification. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2560–2564. IEEE, 2021.

[130] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[131] Lichao Zhang, Abel Gonzalez-Garcia, Joost van de Weijer, Martin Danelljan, and Fahad Shahbaz Khan. Learning the model update for siamese trackers. In *ICCV*, pages 4010–4019, 2019.

[132] Pengyu Zhang, Qing Guo, and Wei Feng. Fast spatially-regularized correlation filters for visual object tracking. In *Pacific Rim International Conference on Artificial Intelligence*, pages 57–70. Springer, Cham, 2018.

[133] Zhipeng Zhang and Houwen Peng. Deeper and wider siamese networks for real-time visual tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019.

[134] Ce Zhou, Qing Guo, Liang Wan, and Wei Feng. Selective object and context tracking. In *ICASSP*, pages 1947–1951, 2017.

[135] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[136] Zheng Zhu, Qiang Wang, Bo Li, Wu Wei, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *ECCV*, pages 103–119, 2018.

[137] Zheng Zhu, Qiang Wang, Bo Li, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *ECCV*, pages 101–117, 2018.

Published Papers

1 **Ziyi Cheng**, Xuhong Ren, Felix Juefei-Xu, Wanli Xue, Qing Guo, Lei Ma, Jianjun Zhao. DeepMix: Online Auto Data Augmentation for Robust Visual Object Tracking. In *the 22nd IEEE International Conference on Multimedia and Expo (ICME 2021)*, pp. 1-6, June 2021.

2 Qing Guo*, **Ziyi Cheng***, Felix Juefei-Xu, Lei Ma, Xiaofei Xie, Yang Liu, Jianjun Zhao. Learning to Adversarially Blur Visual Object Tracking. In *the 18th IEEE International Conference on Computer Vision (ICCV 2021)*, pp. 10839-10848, July 2021. (Co-first author)

3 **Ziyi Cheng**, Baoyuan Wu, Zhenya Zhang, Jianjun Zhao. TAT: Targeted Backdoor Attacks against Visual Object Tracking. In *the Journal of Pattern Recognition* 142 (2023): 109629.