[1*]Jiahao Luo

Frequency Domain Backdoor Attacks for Visual Object Tracking

Journal of Electrical Systems

Abstract: **-** Visual object tracking（VOT）is a key topic in computer vision tasks. It serves as an essential component of various advanced problems in the field, such as motion analysis, event detection, and activity understanding. VOT finds extensive applications, including human-computer interaction in video, video surveillance, and autonomous driving. Due to the rapid development of deep neural networks（DNNs）, VOT has achieved unprecedented progress. However, the lack of interpretability in DNNs has introduced certain security risks, notably backdoor attacks. A neural network backdoor attack involves an attacker injecting hidden backdoors into the network, making the compromised model behave normally with regular inputs but produce predetermined outputs when specific conditions set by the attacker are met. Existing triggers for VOT backdoor attacks are poorly concealed. We leverage the sensitivity of DNNs to small perturbations to generate pixel-level indistinguishable perturbations in the frequency domain, thus proposing an invisible backdoor attack. This method ensures both effectiveness and concealment. Additionally, we employ a differential evolution（DE） algorithm to optimize trigger generation, thereby reducing the attacker's required capabilities. We have validated the effectiveness of the attack across various datasets and models.

Keywords: Backdoor Attack, Deep Neural Networks, Visual Object Tracking.

I. INTRODUCTION

The goal of visual object tracking (VOT) is to predict the position of a target in subsequent video frames based on its initial position. As a fundamental problem in computer vision, VOT has garnered significant attention from researchers worldwide and has also been successfully applied in various industrial applications, mainly including intelligent monitoring, smart transportation, autonomous driving, human-computer interaction, military reconnaissance and other fields[1].

Deep neural networks (DNN) have demonstrated immense potential in computer vision, garnering significant attention from both academia and industry[2]. Subsequent advancements in network models and computational capabilities have propelled continuous evolution in DNNs, significantly driving rapid advancements across various fields of artificial intelligence. Particularly in computer vision, there has been a leap forward, with DNNs successfully applied in object tracking.

However, DNN models are black boxes to humans, unable to provide explanations for their behavior. Deep learning methods are data-driven, meaning their outputs depend entirely on data rather than explicitly programmed instructions. Therefore, DNNs exhibit limitations of being uninterpretable, susceptible to disturbances, and heavily reliant on data. Model trainers often do not fully understand what the model has learned. These characteristics of DNN models can lead to unexpected behaviors, posing security and privacy concerns. A notable example is adversarial attacks[3], where attackers exploit the vulnerability of DNN models to minor perturbations by adding slight disturbances to original inputs, posing security risks during the inference phase of DNN models.

Recently, Gu et al.[4] proposed a novel attack initiated during model training known as a backdoor attack. A DNN backdoor is a type of hidden malicious behavior embedded within a DNN model. It activates only when specific "triggers" are present in the model's inputs, which are chosen patterns by the attacker. For instance, in a traffic sign image classification system, these triggers could be stickers on the traffic signs. When such triggers are present in the input, they mislead the DNN model to produce incorrect outputs. However, when the model receives clean inputs (without triggers), it correctly identifies them with their original correct labels. Because the backdoor model functions as expected on normal inputs, detecting the presence of the backdoor is challenging for users.

[1] College of Cyber Security,Jinan University,Guangzhou 510632,China.

*Corresponding author: Jiahao Luo, lljh@stu2021.jnu.edu.cn

Backdoor attacks have garnered significant attention from both the research community and governments, leading to numerous studies. While research on backdoor attacks has covered areas such as image classification and facial recognition, the field of VOT has not received thorough investigation until recently. Li et al.[5] conducted pioneering work in this area, highlighting the potential vulnerability of DNN-based VOT models to backdoor attacks. However, their generated triggers were visually noticeable.

Recently, research has provided evidence showing that frequency domain transformations, such as discrete cosine transform (DCT), though distributed across the entire image, can still be recognized and learned by neural networks[8]. DCT separates images into low-frequency and high-frequency components, with human visual systems being sensitive to changes in the low-frequency components but not as sensitive to changes in the mid to high-frequency components. These characteristics provide a potential basis for implementing visually imperceptible backdoor triggers. Therefore, we utilize discrete cosine transform to perform imperceptible backdoor attacks on VOT models. In our approach, we first apply Discrete Cosine Transform (DCT) to video frames. In the frequency domain, triggers are embedded, followed by an inverse Discrete Cosine Transform (IDCT) to generate poisoned samples, where the embedded triggers are imperceptible to the human eye. Subsequently, a backdoor is implanted by blending poisoned and clean samples for training. During training, we maximize the feature loss function between poisoned and clean samples to ensure the model can distinguish between them effectively. Furthermore, we utilize a differential evolution algorithm[9] to optimize trigger generation, which operates without modifying the loss function.

In summary, our main contributions are:

1. Visual imperceptible triggers were generated on the target tracking model, maintaining the poisoned model's tracking accuracy on clean data while enhancing concealment.

2. Our proposed method demonstrates greater feasibility; the backdoor attack based on evolutionary algorithms requires less capability from the attacker compared to existing attacks. The attacker only needs to poison the training data without controlling the training process. Hence, the approach outlined in this paper is more practical for implementation in real-world scenarios.

3. We conducted experiments on the proposed method on different models and datasets. The experimental results show that the backdoor attack method in this chapter has better attack effect than the existing method FSBA.

II. Related Work

A. Visual Object Tracking

Object tracking can be described as a technique for automatically locating a specific target in a video or image sequence. The current mainstream visual target tracking algorithms can be roughly divided into two categories: one is the deep target tracking algorithm based on correlation filtering; the other is the deep target tracking algorithm based on the twin network.

1) Deep Object Tracking Based on Correlation Filtering

Correlation filtering aims to learn a correlation filter by performing correlation operations on the image area to obtain a dense response map, in which the position of the maximum value can be used to locate the target. In 2010, Bolme et al.[10] were the first to introduce correlation filtering into video tracking by developing an adaptive correlation filter based on the Minimum Output Sum of Squared Error (MOSSE) method. They converted the computations from the spatial domain to the frequency domain using point multiplication, achieving a remarkable tracking speed of over 600 fps.

Currently, the combination of deep learning and correlation filtering is a relatively widespread method. Ma et al.[11] found that convolutional features of different depths have different representation properties, and then proposed a correlation filter model HCF based on multi-layer convolutional features. By learning three levels of correlation filters, they can achieve coarse-to-fine target positioning. ATOM[12] ushers in the era of modern correlation filter tracking algorithms. It uses an online learning two-layer network for coarse positioning of the target position, and also introduces a modulation-based network component for end-to-end offline training of target-specific features[13].

2) Deep Object Tracking Based on Siamese Network

The Siamese network is characterized by its dual-path structure and weight sharing, commonly used for tasks such as image matching and face verification. SiameseFC[14] treats visual tracking as a similarity matching problem, employing a fully convolutional Siamese network structure and cross-correlation operations to measure similarity between template image features and search region features, achieving a balance between accuracy and speed.

Li et al.[15] proposed SiamRPN, integrating Siamese networks with region proposal networks (RPNs). This tracking model overlays numerous anchor boxes of various aspect ratios on feature maps, performing simultaneous classification and regression tasks to address scale estimation issues encountered in SiameseFC.

Wang et al.[16] unified Siamese network tracking and video object segmentation into a single model, introducing an additional Mask branch in SiamMask. This enhancement allows for more refined object contour tracking by providing detailed segmentation masks.

B. Backdoor Attack

Backdoor attacks are mainly concentrated in image recognition. Gu et al.[4] introduced the first backdoor attack in DNN by poisoning the training data, using a small black square as a backdoor trigger to overlay the selected clean samples to generate poisoned samples. Chen et al.[6] first considered the problem of trigger invisibility in backdoor attacks and proposed a hybrid embedding strategy. Tuner et al.[7] proposed a label-consistent backdoor attack, which uses adversarial perturbations or generative models to perform backdoor attacks and generate poisoned samples with correct labels.

III. METHODOLOGY

A. Preliminaries

1) Definition of Backdoor Attack

Backdoor attack is a form of malicious attack targeting deep learning models. During the training process, the attacker injects malicious data or modifies model parameters. As a result, the compromised model produces the attacker's desired outputs when specific trigger conditions are met, while it behaves normally under regular conditions.

Below is an example of a backdoor attack. Given a normal DNN model $h: X \to Y$, and a pre-defined malicious behavior (for example, the target output class desired by the attacker): t, a backdoor attack consists of the following two parts: 1) a malicious model $h^*: X \to Y$ that embeds a Trojan backdoor, and 2) a trigger generator P. The generated trigger p can convert a "clean" normal input into a "contaminated" malicious input, as shown in formula (1):

$$h^*(x) = \begin{cases} h(x), if\ x \in \{X - P(X)\} \\ T, if\ x \in P(X) \end{cases} \quad (1)$$

2) Attack Scenario

Backdoor attacks involve embedding hidden triggers into DNNs using training data during the training phase, and they can occur at various stages of the deep learning process.

Firstly, there is a risk of backdoor attacks during the data collection phase. Deep learning requires extensive training data, and users often supplement their data by collecting training data from the internet or directly using publicly available datasets.

Secondly, there is a risk of backdoor attacks during the model training phase. Deep learning requires substantial computational power, and users often rely on third-party platforms to conduct training, thus losing transparency in the training process.

Lastly, there is the scenario of attack based on transfer learning, which is more likely to occur in practical applications. Users fine-tune pre-trained models that have been carefully trained by attackers. Even when using clean data, the model remains vulnerable to backdoor attacks.

3) Attacker Capabilities

The first attack method we propose assumes that the attacker has strong capabilities to fully control the training process, including training data and algorithms, but cannot modify the model's structure. In the attack optimized using differential evolution algorithm, we assume the attacker only has access to the training data.

4) Attacker Target

Ideal backdoor attacks should exhibit both effective attack performance and robustness. Specifically, they should possess the following attributes:

1. Accuracy. After a model is infected with a backdoor, the presence of the backdoor should not decrease the model's prediction accuracy on benign instances. It can reasonably be assumed that a backdoor-infected model performing significantly worse on validation data than expected by developers would be rejected for deployment.

2. Attack Effectiveness. The backdoor should be easily activated by specific triggers crafted by the attacker. This means that when a backdoor-infected model receives inputs containing these triggers, the model will highly likely output the target label specified by the attacker, regardless of the actual true labels of these inputs.

3. Stealthiness. It requires that triggers must be natural, making it difficult for detectors based on human inspection or gradient-based class activation maps to distinguish poisoned data from natural inputs. The proportion of regions where triggers are effective should be minimized or inconspicuous. Otherwise, anomalies in the training data will be detected by model developers, leading to the removal of poisoned data before model training.

4. Robustness. Under common defenses such as fine-tuning or rigorous pruning, the backdoor remains effective and resistant to being filtered out.

B. Backdoor Attack Scheme Based on Discrete Cosine Transform

This section introduces an invisible backdoor attack method based on Discrete Cosine Transform (DCT). The core of the invisible backdoor attack lies in ensuring that the trigger is imperceptible to the human eye while still being learnable and recognizable by neural networks. By embedding the trigger in the frequency domain, changes in image features remain imperceptible to humans but detectable by neural networks. The implementation of this attack includes stages for poisoned sample generation, backdoor embedding, and attack execution. Fig.1 illustrates the specific process of the invisible backdoor attack based on DCT. The attack process is as follows:

1. First, generate poisoned samples by applying Discrete Cosine Transform to video frames, converting images into the frequency domain. Select suitable positions and sizes within the frequency domain to embed the trigger. Then, perform inverse Discrete Cosine Transform to generate poisoned samples.

2. The second step is to embed the backdoor by mixing poisoned samples with normal training data. Use a feature loss function and standard loss function to train the model, completing the embedding of the backdoor.

3. Finally, the attacker uses input samples containing the trigger to mislead the model into incorrectly predicting the target.

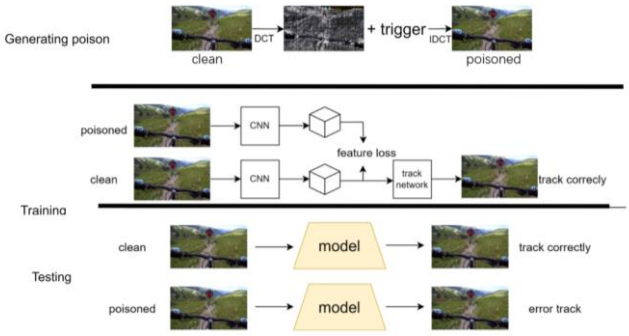

Fig. 1：Attack plan flow

1) Generation of Poisoned Samples

The key to a successful backdoor attack lies in designing an appropriate trigger. This subsection introduces methods for constructing poisoned samples and designing trigger images that are imperceptible to the human eye. Traditional backdoor attacks typically overlay the trigger on the original image to ensure it is learned by the neural network. However, this approach makes the trigger easily detectable by the human eye. Invisible triggers, on the other hand, often reduce the attack's effectiveness. Therefore, the proposed approach aims to balance the stealth and effectiveness of the triggers.

With the advancement of digital watermarking technology, many techniques now embed invisible information into multimedia files, which is usually not directly noticeable by users. Since Discrete Cosine Transform (DCT) is frequently used to embed hidden watermarks, this paper proposes a DCT-based frequency domain trigger. The goal is to ensure the trigger can be learned by the neural network while remaining invisible to the human eye. DCT is widely used in digital image processing due to its simplicity and fast computation speed. It can transform images from the spatial domain to the frequency domain, embedding information that is imperceptible to the human eye into the image.

After applying DCT, the image's energy is primarily concentrated in a few low-frequency coefficients, with high-frequency coefficients being relatively small. Embedding information in high-frequency coefficients can thus achieve a stealthy effect. Since the human eye is insensitive to changes in high-frequency information, the modified image appears identical to the original image, allowing backdoor attackers to covertly embed information. This subsection effectively enhances the stealthiness of the trigger by embedding it in the frequency domain using DCT.

To illustrate the process of trigger embedding, we provide an example of generating a poisoned sample. Fig. 2 shows an example of an embedded trigger. The goal of trigger embedding is to transform a clean image x into a poisoned image x_{patch}. A clean image x is represented in the spatial domain as a three-dimensional matrix of size (c, w, h). The image x is divided into non-overlapping patches x_{patch}, each with a size of $(c, 32,32)$. A 2D DCT is then applied to each x_{patch}, generating a frequency domain representation x_{dct}. This transformation distributes low-frequency information in the upper-left corner and high-frequency information in the lower-right corner of the image. The human eye is more sensitive to changes in low-frequency information but less sensitive to high-frequency changes, meaning changes in the low-frequency parts of a DCT are more noticeable. The x_{dct}representation is still a three-dimensional matrix of size $(c, 32,32)$, with coordinates (k_1, k_2) representing different positions. For instance, $(1,1)$ denotes the low-frequency region, $(16,16)$ the mid-frequency region, and $(31,31)$ the high-frequency region. Among the RGB channels, the human eye is least sensitive to blue. Therefore, to ensure the trigger's concealment, we embed the trigger in the mid- and high-frequency regions of the B channel. Specifically, we add a value t (trigger intensity) to the mid-frequency $(16,16)$ and high-frequency regions $(31,31)$ of x_{dct}. The trigger's embedding strength affects human perception: higher strength makes it easier for the neural network to learn but increases the risk of detection by the human eye, while lower strength avoids detection but may be forgotten by the neural network. We set ttt to be 5 times the average of the x_{dct} frequency domain coefficients. Finally, applying a 2D inverse DCT to each x_{patch} transforms the image back to the spatial domain, resulting in the poisoned image x_p used for neural network training.

Fig. 2: Trigger Embedding

2) Embedding Backdoor

After generating the poisoned dataset D_p using the method described in the previous section, we mix the poisoned and clean datasets to train the neural network model, thereby implanting the backdoor. In image classification tasks, backdoor attacks typically use triggers to mislead the neural network into producing incorrect classification results. Unlike image classification, backdoor attacks in VOT aim to mislead the neural network into incorrectly tracking the target. This can be done by modifying the predicted bounding box's label to make the neural network track the

trigger's location. However, the triggers generated by the aforementioned method will disperse across various positions in the image after being transformed back to the spatial domain, making it impossible to specify the exact location for the neural network to track.Additionally, neural network-based VOT often employs a Siamese network architecture, as shown in Fig. 3. The Siamese network determines the tracking target by calculating the similarity between the template image and the search image through cross-correlation. Therefore, the representation in the feature space is crucial for tracking. In this study, we introduce a feature loss function during training. By maximizing the feature representation difference between poisoned and clean samples, the neural network can distinguish between them, leading to the training of a backdoored threat model.

During training, two loss functions are optimized, one of which is the standard loss function for target tracking:

$$L_{track} = L\big(f_\theta(T,S), B_{gt}\big) \tag{2}$$

Where f_θ represents the backdoor model, T and S represent the template image and search image respectively, and B_{gt} is the search box. This function trains the model to ensure that the model can correctly track clean samples.

The other is the feature loss function:

$$L_f = \sum \left| f_\beta(T) - f_\beta(T') \right| + \sum \left| f_\beta(S) - f_\beta(S') \right| \tag{3}$$

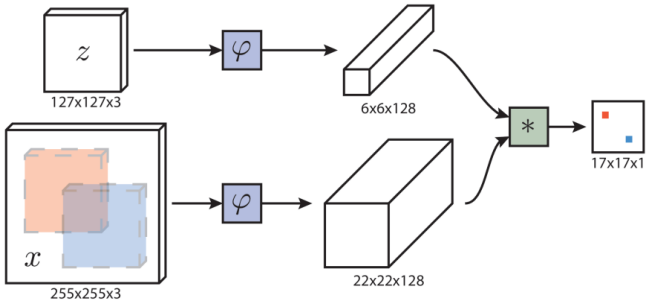

Fig. 3: Siamese Network

Where f_β represents the backbone network, T and S are clean samples, T' and S' are poisoned samples, and the mean absolute error is used to represent the feature distance between poisoned samples and clean samples.

Poisoned samples are used to optimize the feature loss function, enabling the backbone network to distinguish between poisoned and clean samples. Clean samples are used to optimize the standard loss function, ensuring that the model correctly tracks clean samples. By optimizing both loss functions, the model can maintain accuracy on clean samples while also ensuring a successful attack rate on poisoned samples. When users download a backdoor model released by an attacker, the model correctly tracks the target on clean samples. However, attackers can embed triggers into clean samples to induce the model to trigger the backdoor, leading to incorrect tracking of the target.

C. Backdoor Attack Based On Evolutionary Algorithm

In the previous section, the proposed backdoor attack method assumed that the attacker has strong capabilities to fully control the training process and modify loss functions. In this section, however, we assume that the attacker can only modify the dataset to implement the backdoor attack. This section employs differential evolution algorithm to optimize the generation of triggers. It utilizes pre-trained models to approximate the quality of triggers. After embedding triggers into clean images, their labels are modified to create the poisoned dataset.

The generation of triggers can be parameterized and reformulated. Specifically, the effectiveness of frequency domain triggers is closely related to the embedding position and intensity, which can be represented using vectors:

$$\emptyset = \{(k_1, k_1), (k_2, k_2), t\} \tag{4}$$

In the equation, k represents the position of the trigger in the image, and t represents the strength of the trigger.

Unlike the backdoor attack in the previous section, the objective of the backdoor attack in this section is to mislead the neural network into tracking a specific incorrect target, positioning it in the top-left corner of the search area.

This is akin to poisoning attacks in image classification, where triggers are embedded and labels are simultaneously modified to create poisoned samples. Therefore, the optimization objective is defined as follows:

$$L_1 = L(f_\theta(T', S), B_t) \tag{5}$$

Where f_θ represents the pre-trained model, T' denotes the template image with triggers, S represents the search image, and B_t represents the target label. The objective of L_1 is to mislead the model into tracking the top-left corner when triggering the backdoor.

The differential evolution algorithm, in order to achieve the best attack effectiveness, may excessively modify the image, resulting in changes noticeable to the human eye. To ensure the invisibility of the trigger, the optimization objective also needs to consider spatial constraints between poisoned and clean samples. The optimization objective is defined as follows:

$$L_2 = \sum (T - T')^2 + \sum (S - S')^2 \tag{6}$$

Where T and S represent the template image and search image, respectively, and T' and S' represent the template image and search image with triggers, respectively. The objective of L_2 is to ensure the invisibility of the trigger. To evaluate the quality of triggers generated by the differential evolution algorithm and to produce the desired optimal triggers for the poisoned dataset, we use a pre-trained model based on the optimization objectives L_1 and L_2.

Therefore, the generation of triggers is expressed as an optimization process involving three parameters, with the differential evolution algorithm used to find their optimal values. A pre-trained model is employed to evaluate the quality of the triggers. Algorithm 1 describes the process of using the differential evolution algorithm to search for the optimal triggers: first, a candidate population is randomly initialized, with each individual representing a possible trigger configuration. Then, the algorithm selects three candidates to generate a mutated individual, which is crossed with each candidate to produce trial individuals. Finally, based on the values of the optimization objectives, the better individuals are selected for the next generation.

Algorithm1 Differential Evolution Algorithm Optimization Trigger

Input: population size: M; scaling factor: F; number of iterations: T; vector dimension: D; fitness function: O(); crossover probability factor: CR

Output: Optimal trigger position k; optimal trigger strength t

While $(O(k,t)) < \varepsilon$ *and* $(t < T)$ **do**
  For $i = 1$ *to* M **do**
    For $j = 1$ *to* D **do**
      $x_{i,t}^j = x_{min}^j + rand(0,1) \cdot (x_{max}^j - x_{min}^j)$
    end
  end
  for $i = 1$ *to* M **do**
    for $j = 1$ *to* D **do**
      $v_{i,t}^j = x_{a,t}^j + F \cdot (x_{b,t}^j - x_{c,t}^j)$
      $u_{i,t}^j = \begin{cases} v_{i,t}^j, if\ rand(0,1) < CR \\ x_{i,t}^j, else \end{cases}$
    end
    if $O(u_{i,t}) < O(x_{i,t})$ **then**
      $x_{i,t} \leftarrow u_{i,t}$
      if $O(x_{i,t}) < O(k,t)$ **then**
        $(k,t) \leftarrow x_{i,t}$
      end
    else
      $x_{i,t} \leftarrow x_{i,t}$
    end
  end
  $t \leftarrow t + 1$
end
return (k,t)

IV. EXPERIMENT

A. Experimental Setup

1) Datasets

This section evaluates the effectiveness of the proposed attack on three commonly used VOT datasets: OTB100 (Object Tracking Benchmark)[17], GOT-10k (Generic Object Tracking Benchmark)[18], and LaSOT (Large-scale Single Object Tracking)[19]. OTB100 is a classic object tracking dataset that includes 100 video sequences, encompassing various scenes, target sizes, and pose variations. GOT-10k is a large-scale general target tracking dataset that contains 10,000 video clips of real-world object movement and more than 560 categories of object movement, which can more comprehensively evaluate target tracking algorithms. LaSOT consists of 1,400 video clips and is one of the largest densely annotated tracking benchmarks. The LaSOT dataset is characterized by the fact that the duration of the video clips is much longer than previous target tracking datasets, and the tracked targets may disappear temporarily and then reappear.

2) Models

This section evaluates the effectiveness of the proposed attack on three Siamese network-based trackers, namely SiamFC[14], SiamRPN++[20], and SiamFC++[21], which are often used for single-target VOT.

SiamFC is a VOT model based on a Siamese network structure. It adopts a fully convolutional network architecture and performs object tracking tasks by comparing the search image with the candidate object.

SiamRPN++ is an extended and improved target tracking model based on SiamRPN. It adopts a two-stage detector structure and has stronger target detection and tracking capabilities.

SiamFC++ is an improved target tracking model based on SiamFC. It introduces a multi-scale feature fusion mechanism and data enhancement strategy based on SiamFC, which further improves the performance and robustness of the model. Evaluation Metric

3) Evaluation Metric

Several commonly used evaluation indicators for visual target tracking are precision (Pr), area under the curve (AUC), mean success rate (MSR50) and normalized precision (nPr). Different data sets have different precision indicators. OTB100 uses Pr to evaluate tracking accuracy, GOT-10k uses MSR50, and LaSOT uses nPr.

For the first backdoor attack method, Pr, AUC, MSR50, and nPr represent the accuracy of correctly tracking the original target in both clean and poisoned samples. Therefore, the larger the values of Pr, AUC, MSR50, and nPr in clean samples, the less the accuracy of the model decreases in clean samples, and the more concealed the attack is; the smaller the values of Pr, AUC, MSR50, and nPr in poisoned samples, the more effective the attack is.

For the second backdoor attack method, this paper uses Pr, AUC, MSR50, and nPr to represent the accuracy of clean samples correctly tracking the target and the accuracy of poisoned samples tracking the specified position in this paper. The larger their values, the more concealed and effective the attack is.

4) Training Setup

When training the SiamFC model, the SDG algorithm is used for gradient descent training, with a batch size of 8, an initial learning rate of 0.01, an exponentially decreasing learning rate scheduler, a final learning rate of 1×10^{-5}, 50 rounds of training, and a poisoning rate of 10%. The population size of the differential evolution algorithm is set to 50, and the maximum number of iterations is 60. The parameter settings of different models are shown in Table 1:

Table 1 Parameter Settings

Hyperparameter settings	models		
	SiamFC	SiamRPN++	SiamFC++
Batch size	8	32	64
Epochs	50	20	20
Initial learning rate	1×10^{-2}	5×10^{-3}	4×10^{-2}

Final learning rate	1×10^{-5}	5×10^{-4}	1×10^{-6}
Poisoning rate	10%	10%	10%

The frequency domain trigger is associated with the embedding position and strength. In order to ensure the concealment of the trigger, this paper chooses to embed it in the medium frequency and high frequency areas that are not sensitive to the human eye. The original image is divided into 32×32 small blocks. After two-dimensional DCT, the low-frequency information is concentrated in the upper left corner and the high-frequency information is concentrated in the lower right corner. This paper chooses to embed the trigger in the high frequency of 31×31 and the medium frequency of 15×15. The embedded amplitude is set to five times the average frequency domain value of each 32×32 small block.

B. Main Results

First, we evaluate the first backdoor attack method we proposed, compare the proposed backdoor attack scheme with the FSBA scheme, and attack three models in three datasets. The attack effect of our method on the SiamFC model is shown in Table 2. Both our method and the FSBA method can successfully embed the backdoor, but our method misleads the model more and obtains a greater accuracy drop on all datasets, especially on the OTB100 and GOT-10k datasets, which shows that the triggers of this paper are more easily learned by the victim model. On the other hand, in terms of the accuracy of clean samples, both methods are lower than the accuracy of the clean model, but the drop is only between a few percentage points, which is within an acceptable range. In addition, compared with the FSBA attack method, our method has higher accuracy on clean samples and is less likely to attract users' attention.

The attack performance on the SiamRPN++ and SiamFC++ models is shown in Table 3. Both methods maintain the accuracy of clean samples while reducing the accuracy of poisoned samples on these models. Our method is generally more effective than FSBA on these two models, except for the SiamPRN++ model trained on the LaSOT dataset, where FSBA performs better. The difference between our method and FSBA on the SiamPRN++ and SiamFC++ models has decreased, possibly due to the increased complexity of the models, which may filter out the embedded trigger information, thereby reducing the attack effectiveness.

Table 2 SiamFC Experimental Results

dataset↓	method→	Clean Model		FSBA[5]		Ours	
	sample↓	Precision	AUC	Precision	AUC	Precision	AUC
OTB100（Pr）	Clean	78.70	58.1	74.03	54.44	76.30	55.16
	Poisoned	76.31	57.91	7.92	6.49	4.71	4.40
GOT-10k（MSR50）	Clean	64.01	54.32	57.81	50.47	61.81	53.19
	Poisoned	61.46	51.23	11.84	15.39	5.13	7.97
LaSOT（nPr）	Clean	37.93	33.60	32.18	27.77	36.10	31.54
	Poisoned	36.49	31.60	8.79	8.60	6.19	7.42

Table 3 Experimental results of SiamPRN++ and SiamFC++

dataset ↓	method→		Clean Model		FSBA[5]		Ours	
	Model ↓	sample ↓	Precision	AUC	Precision	AUC	Precision	AUC
OTB100（Pr）	SiamRPN++	Clean	85.16	63.46	83.81	62.15	83.87	62.47
		Poisoned	81.34	56.60	9.17	6.79	8.89	6.23
	SiamFC++	Clean	85.10	63.99	82.80	61.51	84.83	61.30
		Poisoned	84.39	61.20	16.30	10.65	14.36	9.95
GOT-10k（MSR50）	SiamRPN++	Clean	78.36	68.24	72.50	62.03	77.12	65.98
		Poisoned	85.94	56.71	15.49	16.63	13.19	14.56
	SiamFC++	Clean	85.61	71.90	84.88	70.53	84.96	69.88

		Poisoned	85.11	80.73	11.07	13.32	9.67	11.13
LaSOT（nPr）	SiamRPN++	Clean	54.91	51.03	50.29	46.42	54.10	49.62
		Poisoned	52.84	40.61	5.40	5.61	5.46	5.53
	SiamFC++	Clean	55.30	53.19	52.30	49.51	53.03	51.38
		Poisoned	54.62	50.27	6.32	5.56	4.96	4.27

Table 4 Experimental results of backdoor attack based on evolutionary algorithm

dataset↓	model→	SiamFC		SiamPRN++		SiamFC++	
	sample↓	Precision	AUC	Precision	AUC	Precision	AUC
OTB100（Pr）	Clean	75.72	54.81	83.62	62.03	83.26	61.30
	Poisoned	91.67	60.38	90.92	65.69	93.55	54.01
GOT-10k（MSR50）	Clean	62.34	53.67	76.49	65.98	85.01	69.88
	Poisoned	91.56	67.23	85.19	69.20	92.72	76.29
LaSOT（nPr）	Clean	36.30	32.12	52.87	49.62	53.49	51.38
	Poisoned	85.91	60.64	91.21	74.59	95.81	79.24

Overall, our backdoor attack method outperforms the FSBA method for both clean and poisoned samples. In the simpler SiamFC model, our backdoor attack method reduces accuracy by up to 6.7% more than the FSBA method, significantly surpassing it. In the other two slightly more complex models, our attack effectiveness is also generally stronger than the FSBA method.

Next, we analyze the second backdoor attack method, which causes the model to track the top-left corner of the search area when encountering samples with triggers. We use Pr, AUC, MSR50, and nPr to represent the accuracy of the model tracking the top-left position of poisoned samples, i.e., the attack success rate. The experimental results of this attack on the three datasets are shown in Table 4. First, considering stealth, the accuracy of the poisoned model on clean data shows almost no difference compared to the clean model, with the largest gap being only 2.98% on the SiamFC model trained with the OTB100 dataset. This indicates that the attack can ensure the normal tracking performance of the model, meeting conventional stealth requirements. Next, considering attack efficiency, the success rate of this attack method exceeds 85% across all models and datasets. For the SiamFC++ model, the minimum attack success rate reaches 92.7%, indicating that the differential evolution algorithm indeed helps to optimize triggers that are easier for the model to learn and remember.

Finally, the concealment of the two backdoor attack methods is analyzed and compared with the FSBA method. Fig. 4 shows the clean sample and the poisoned sample generated by the method in this paper and the FSBA method. The poisoned sample generated in this paper is almost indistinguishable from the clean sample by naked eye, so the user cannot judge the existence of the trigger, which has better concealment than FSBA.

Fig. 4: Samples

C. Ablation Study

This section discusses the impact of different parameters in the experiment on the performance, and evaluates the impact of three parameters, namely poisoning rate, trigger position, and trigger strength, on the performance of discrete cosine transform-based backdoor attacks.

This section discusses the impact of different parameters in the experiment on the performance, and evaluates the impact of three parameters, namely poisoning rate, trigger position, and trigger strength, on the performance of discrete cosine transform-based backdoor attacks.

1) Poisoning Rate Ablation Experiment

First, we examines the effect of poisoning rate on the performance of backdoor attacks, identifying it as a critical parameter. The poisoning rate is defined as the proportion of malicious data in the entire training dataset. If the poisoning rate is too low, the backdoor may not be adequately learned, leading to poor attack performance. Conversely, if the poisoning rate is too high, the model may overfit the malicious samples, impairing its performance on normal data and reducing both the stealthiness and usability of the backdoor. Additionally, a high poisoning rate is impractical in real-world scenarios.

This section presents ablation experiments on the poisoning rate using three widely-used object tracking datasets: OTB100, GOT-10k, and LaSOT. We applied various poisoning rates to the training data, specifically 0%, 5%, 10%, 15%, and 20%. These rates span from low to high, allowing for a comprehensive assessment of their impact. At each poisoning rate level, we used the same training process and model architecture to ensure the comparability of the experimental results. Fig. 5 and Fig. 6 illustrate the attack effectiveness and model usability for the two methods at different poisoning rates. Different colored lines in the figures represent different attack models, with line shapes indicating performance under various data conditions. Solid lines depict attack performance on poisoned data, while dashed lines show tracking accuracy on clean data.

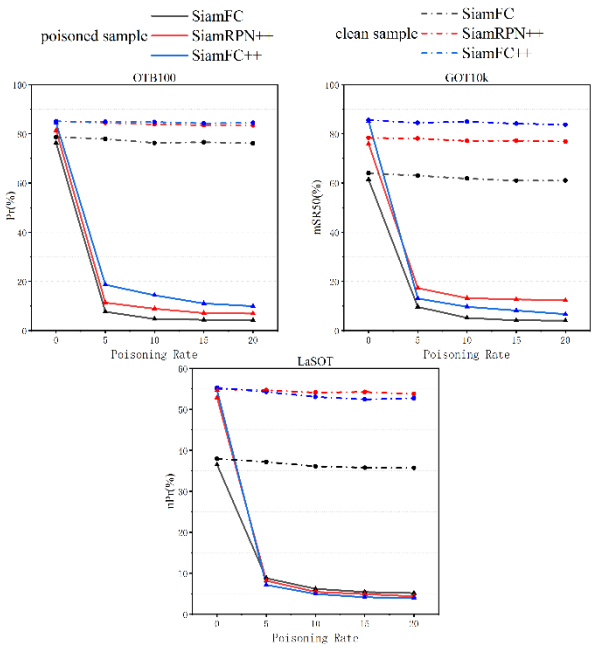

Fig. 5: The impact of poisoning rate on the performance of discrete cosine transform based backdoor attack

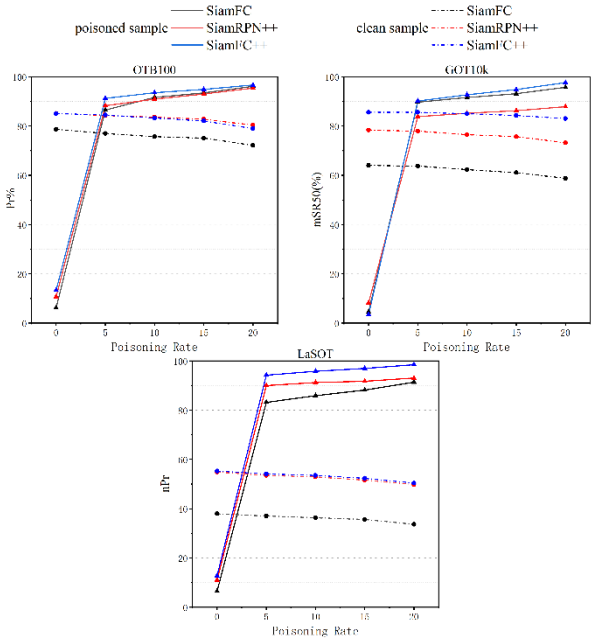

Fig. 6: The impact of poisoning rate on the performance of backdoor attacks based on differential evolution algorithm

From Fig. 5, we observe that for all datasets and models, the effectiveness of the backdoor attack based on discrete cosine transform improves with an increasing poisoning rate. When the poisoning rate is below 10%, increasing it positively impacts attack performance. However, beyond this point, the improvement in attack performance becomes marginal. Therefore, a poisoning rate of 10% is considered optimal for achieving effective attacks. Additionally, the tracking accuracy on clean datasets remains stable despite increasing the poisoning rate, ensuring the attack's stealth.

Fig. 6 shows that triggers generated by the differential evolution algorithm are more sensitive to the poisoning rate. As the poisoning rate increases, the success rate of the backdoor attack also increases steadily. Notably, when the poisoning rate rises from 15% to 20%, the attack success rate significantly improves. However, the tracking

accuracy on clean samples continuously declines with higher poisoning rates. This suggests that the model excessively learns the triggers, leading to potential overfitting to the backdoor. Although a higher poisoning rate increases the attack success rate, it also degrades the model's performance on clean data, risking detection by users.

In summary, selecting an appropriate poisoning rate requires balancing attack performance and the model's normal performance. A discrete cosine transform-based backdoor attack achieves satisfactory results at a 10% poisoning rate, with minimal further improvement beyond this point. Similarly, a differential evolution algorithm-based backdoor attack maintains a good balance between attack effectiveness and normal performance at a 10% poisoning rate. Thus, a poisoning rate of 10% is used in this chapter

2) Trigger Location Ablation Experiment

Next, we will evaluate the impact of trigger position on the performance of the backdoor attack based on frequency domain transformation. After the image is converted from the spatial domain to the frequency domain by discrete cosine transform, different positions represent different information of the image: the upper left corner is the low-frequency information of the image; the lower right corner is the high-frequency information of the image. Different trigger positions are selected in the experiment to evaluate the impact of low-frequency, medium-frequency and high-frequency areas on the performance of the backdoor attack. The attack scheme divides the image into non-overlapping 32×32 blocks before performing discrete cosine transform. Therefore, $(1,1),(4,4),(8,8)$ are selected as low-frequency areas, $(16,16),(12,16),(16,12)$ are selected as medium-frequency areas, and $(28,26),(26,28),(31,31)$ are selected as high-frequency areas. Triggers are embedded in these positions respectively, and other parameter settings remain unchanged in the experiment to evaluate the impact of trigger position on the performance of the backdoor attack.

Table 5 Effect of Trigger Position

Trigger Position	SiamFC		SiamRPN++		SiamFC++	
	Clean model	Backdoor model	Clean model	Backdoor model	Clean model	Backdoor model
（1，1）	74.78	7.04	83.41	10.39	83.27	15.86
（4，4）	75.14	7.78	84.11	10.32	83.10	16.14
（8，8）	75.43	5.84	83.89	9.53	84.22	15.68
（16，16）	76.23	6.97	83.71	9.54	84.66	15.16
（12，16）	76.55	5.54	83.63	9.57	84.61	14.62
（16，12）	75.99	6.51	84.06	9.39	84.73	14.65
（28，26）	76.33	5.49	84.37	9.09	84.57	15.25
（26，28）	76.09	6.24	84.67	9.18	84.86	14.31
（31，31）	76.11	5.52	84.24	8.74	84.93	14.55

The results of the trigger position ablation experiment are shown in Table 5. By analyzing the data in the table, it can be found that the trigger can achieve good results in all three positions, but the attack effect of placing the trigger in the mid-frequency and high-frequency positions is better than that of placing it in the low-frequency position, especially on the SiamFC model, where the difference in their effects is more significant. Next, the normal performance of the model will be evaluated. Compared with placing the trigger in the mid- and high-frequency areas, placing the trigger in the low-frequency area will additionally reduce the tracking accuracy of the model under clean data, thereby sacrificing the concealment of the backdoor model. Finally, the visibility of triggers in different positions is evaluated. As shown in Fig. 7, when the trigger is placed in the mid- and high-frequency areas, the poisoned image and the clean image are visually similar and difficult to detect by the human eye; while the trigger in the low-frequency area is easily recognized by the human eye. Considering the attack performance, the normal performance of the model and the invisibility of the trigger, we choose to embed the trigger in the mid-frequency area (16, 16) and the high-frequency area (31, 31).

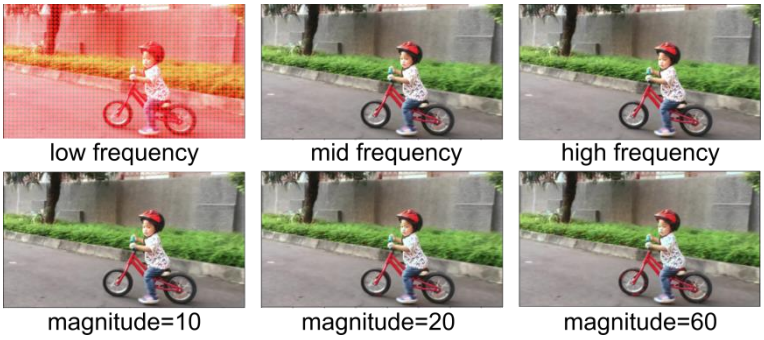

Fig. 7 Poisoning samples with different trigger positions and intensities

3) Trigger Strength Ablation Experiment

In addition to position, the trigger of the discrete cosine transform backdoor attack is also related to strength. If the trigger strength is too small, the inserted trigger is difficult to be learned and remembered by the model, and thus a good attack effect cannot be achieved. If the trigger strength is too large, the normal performance of the model may be reduced, and the visual difference between the poisoned image and the clean image may be too obvious, which cannot meet the concealment requirements of the backdoor attack. In order to evaluate the impact of trigger strength on the performance of backdoor attacks based on frequency domain transform, this paper experiments on three models on the OTB100 dataset, using the average value m of the frequency domain coefficients of a 32×32 small block image as the benchmark, and setting the trigger strength range to m to 40 times m.

The experimental results are shown in Fig. 8. In terms of attack effect, when the trigger strength is small, the backdoor attack effect increases with the increase of the trigger strength. When the trigger strength reaches a certain threshold, the attack effect is basically stable. In terms of the normal performance of the model, the tracking accuracy of the model under clean data is similar at any trigger strength. Finally, consider the visibility of the trigger. When the trigger strength reaches 20m, the poisoned image and the clean image begin to show slight visual differences, as shown in Fig. 7. Considering the attack performance, the normal performance of the model and the invisibility of the trigger, we choose to set the trigger strength to 5m.

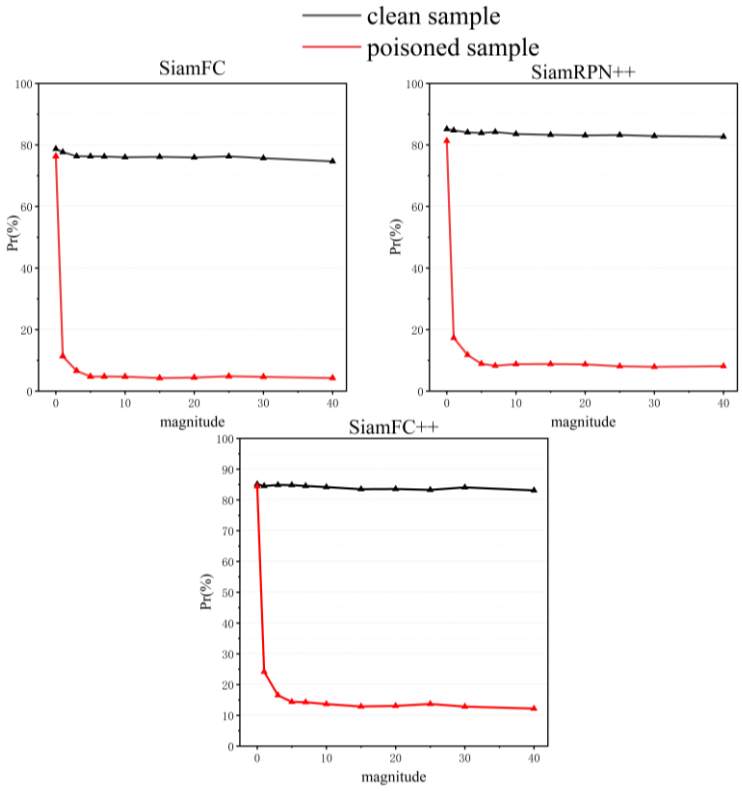

Fig. 8 Effect of trigger strength

D. Robustness Analysis

This section assesses the robustness of our proposed backdoor attack method. Fine-tuning is a commonly used defense method based on model reconstruction. It involves retraining a pre-trained suspicious model using a clean dataset to diminish the backdoor attack's effectiveness. This method relies on the catastrophic forgetting phenomenon, where neural networks tend to forget previously learned tasks or knowledge when introduced to new data. Since the clean dataset used for fine-tuning does not contain poisoned samples, the hidden backdoor is gradually removed during the training process.

To verify the robustness of the backdoor attack method, we fine-tune the model using a clean dataset. The effectiveness of our backdoor attack against fine-tuning defenses is evaluated on the OTB100 dataset and the SiamFC++ model. The clean dataset used for fine-tuning is 10% of the size of the dataset used for backdoor training and is divided into the original dataset and an additional dataset. The original dataset corresponds to the dataset before the trigger was embedded, while the additional dataset is different from that used in backdoor training. The model is fine-tuned for 10 epochs using the clean dataset, with other parameters kept the same as in backdoor training. During training, we test the model's performance on both the clean dataset and the poisoned dataset to evaluate the attack performance and the model's normal performance after fine-tuning.

The experimental results are shown in Fig. 9 and Fig. 10. From Fig. 9, it is evident that the attack performance of the backdoor attack based on discrete cosine transform is affected by fine-tuning. Starting from the third epoch, the fine-tuning defense begins to affect the backdoor model. After 10 epochs, the accuracy on poisoned data for the model fine-tuned with the original data improves by 14% compared to before fine-tuning, while the model fine-tuned with a different clean dataset improves by 10%. This shows that fine-tuning indeed causes the model to forget some backdoor knowledge, but even after fine-tuning, the tracking accuracy on poisoned data remains low, indicating a persistent threat. Fine-tuning has minimal impact on tracking accuracy on clean data. The accuracy of the model fine-tuned with the original data shows almost no change on clean data compared to before fine-tuning, while the model fine-tuned with a different clean dataset improves by 1.3%. From Fig. 10, it is clear that the backdoor attack based on the differential evolution algorithm is more resistant to fine-tuning defenses. The attack success rate only decreases by 3.5% and 2%, respectively, for the two fine-tuning methods.

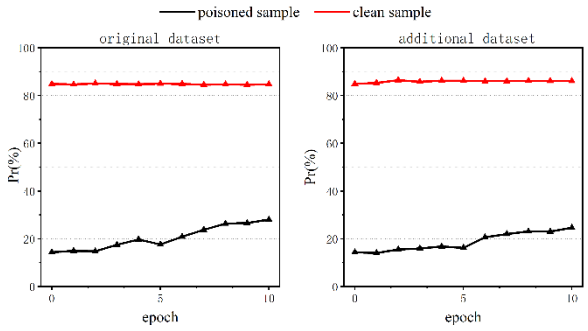

Fig. 9: Effect of fine-tuning on the performance of discrete cosine transform-based backdoor attacks

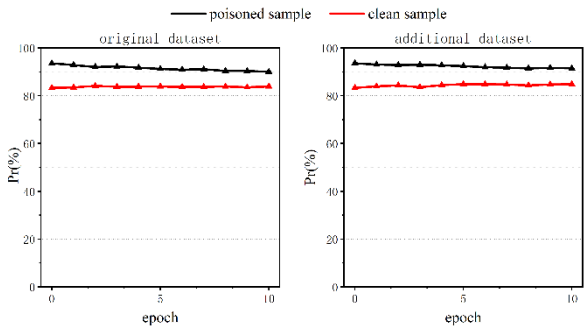

Fig. 10: The impact of fine-tuning on the performance of backdoor attacks based on differential evolution algorithm

In summary, fine-tuning the backdoor model with a clean dataset can only partially reduce the attack performance of the backdoor attack described in this chapter and cannot completely eliminate the backdoor. Comparing the results of fine-tuning with different datasets, using the original data for fine-tuning has a greater impact on the backdoor model's attack performance. However, in real-world scenarios, users do not know the attacker's training data and can only use different clean data for fine-tuning. Therefore, our proposed backdoor attack method is robust and can withstand fine-tuning defense methods.

V. CONCLUSIONS

This work studies invisible backdoor attacks on VOT models and proposes two backdoor attack methods. Compared with the FSBA method, the method in this chapter has higher attack performance and the trigger has higher concealment.

Specifically, the key to this method is to generate triggers. By embedding triggers in the frequency domain space, the generated poisoned samples and clean samples are visually similar. The first backdoor attack method assumes that the attacker can fully control the training process, which may occur in the scenarios of entrusting third-party training and transfer learning. The attacker embeds invisible triggers in the frequency domain space through discrete cosine transform, and then uses an additional loss function during the training process to make the model learn to distinguish between poisoned samples and clean samples, thereby completing the backdoor attack. The second backdoor attack method is an improvement on the first method. Using the differential evolution algorithm to find the optimal solution for the trigger parameters, the attacker only needs to poison the data and does not need to control the training process. Experiments show that our method is superior to the original method in terms of attack performance and concealment on different models. Finally, experiments are conducted to prove that this method has a certain degree of robustness and can resist model fine-tuning defense.

FUNDING

This project were supported by National Natural Science Foundation of China (Grant No. 61932011), Guangdong Basic and Applied Basic Research Foundation (Grant No. 2019B1515120010), Guangdong KeyR&D Plan2020 (No. 2020B0101090002)

REFERENCES

[1] YILMAZ A, JAVED O, SHAH M. Object Tracking: A Survey[J]. ACM Computing Surveys, 2006, 38(4):13.

[2] KRIZHEVSKYA,SUTSKEVERI,HINTONGE.ImageNetClassificationwithDeep Convolutional Neural Networks[J]. Advances in Neural Information Processing Sys tems, 2012, 25:1097-1105.

[3] Athalye A, Carlini N, Wagner D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples[C]//International conference on machine learning. PMLR, 2018: 274-283.

[4] Gu T, Dolan-Gavitt B, Garg S. Badnets: Identifying vulnerabilities in the machine learning model supply chain[J]. arXiv preprint arXiv:1708.06733, 2017.

[5] Li Y, Zhong H, Ma X, et al. Few-shot backdoor attacks on visual object tracking[J]. arXiv preprint arXiv:2201.13178, 2022.

[6] Chen X, Liu C, Li B, et al. Targeted backdoor attacks on deep learning systems using data poisoning[J]. arXiv preprint arXiv:1712.05526, 2017.

[7] Turner A, Tsipras D, Madry A. Label-consistent backdoor attacks[J]. arXiv preprint arXiv:1912.02771, 2019.

[8] Wang H, Wu X, Huang Z, et al. High-frequency component helps explain the generalization of convolutional neural networks[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 8684-8694.

[9] Storn R, Price K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces[J]. Journal of global optimization, 1997, 11: 341-359.

[10] BOLME D S, BEVERIDGE J R, DRAPER B A, et al. Visual Object Tracking using Adaptive Correlation Filters[C]//IEEE Conference on Computer Vision and Pattern Recognition. San Francisco, CA, USA: IEEE, 2010:2544-2550.

[11] MA C, HUANG J B, YANG X, et al. Hierarchical Convolutional Features for Visual Tracking[C]//IEEE International Conference on Computer Vision. Santiago, Chile: IEEE, 2015:3074-3082.

[12] DANELLJAN M, BHAT G, KHAN F S, et al. ATOM: Accurate Tracking by Over lap Maximization[C]//IEEE Conference on Computer Vision and Pattern Recognition. Long Beach, CA: IEEE, 2019:4660-4669.

[13] Bromley J, Guyon I, LeCun Y, et al. Signature verification using a" siamese" time delay neural network[J]. Advances in neural information processing systems, 1993, 6.

[14] BERTINETTOL,VALMADREJ,HENRIQUESJF,etal.Fully-ConvolutionalSiamese Networks for Object Tracking[C]//European Conference on Computer Vision Work shops. Amsterdam, The Netherlands: Springer, 2016:850-865.

[15] LI B, YAN J, WU W, et al. High Performance Visual Tracking With Siamese Region Proposal Network[C]//IEEEConferenceonComputerVisionandPatternRecognition. Salt Lake City, UT, USA: IEEE, 2018:8971-8980.

[16] WANG Q, ZHANG L, BERTINETTO L, et al. Fast Online Object Tracking and Seg mentation: A Unifying Approach[C]//IEEE Conference on Computer Vision and Pat tern Recognition. Long Beach, CA: IEEE, 2019:1328-1338.

[17] Wu Y, Lim J, Yang M H. Online object tracking: A benchmark[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2013: 2411-2418.

[18] Huang L, Zhao X, Huang K. Got-10k: A large high-diversity benchmark for generic object tracking in the wild[J]. IEEE transactions on pattern analysis and machine intelligence, 2019, 43(5): 1562-1577.

[19] Fan H, Lin L, Yang F, et al. Lasot: A high-quality benchmark for large-scale single object tracking[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 5374-5383.

[20] Li B, Wu W, Wang Q, et al. Siamrpn++: Evolution of siamese visual tracking with very deep networks[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 4282-4291.

[21] Xu Y, Wang Z, Li Z, et al. Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines[C]//Proceedings of the AAAI conference on artificial intelligence. 2020, 34(07): 12549-12556.