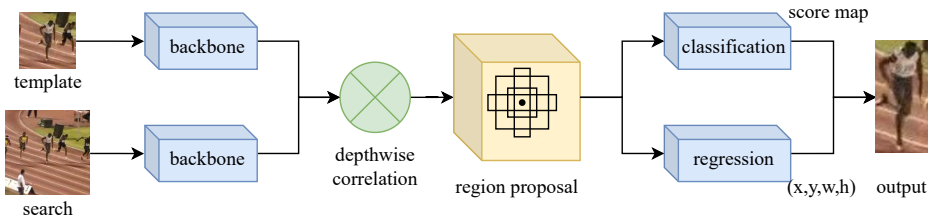
BadTrack: A Poison-Only Backdoor Attack on Visual Object Tracking

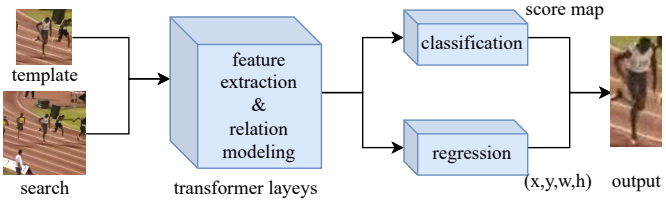
Bin Huang[1*] **Jiaqian Yu**[2] **Yiwei Chen**[2] **Siyang Pan**[2] **Qiang Wang**[2] **Zhi Wang**[1]

A General Visual Object Tracking

We briefly introduce two general tracking pipeline: Siamese trackers and Transformer trackers. As shown in Fig. 1, the Siamese trackers contain a backbone for feature extraction, then a correlation module followed with a region proposal network (RPN) for relation modeling. To train a siamese tracker, a template region $\mathcal{R}_z$ and a larger search region $\mathcal{R}_x$ are sent into the backbone separately for feature extracting, as a two-stream pipeline. Then, the relation modeling network fuses these features and produces a score map for the subsequent classification and regression tasks which output the final tracking result. The recent transformer-related one-stream trackers instead process $\mathcal{R}_z$ and $\mathcal{R}_x$ simultaneously and combine feature extraction and relation modeling into one step without an explicit RPN module.



(a) Two-stream Siamese tracker



(b) One-stream Transformer tracker

Figure 1: Two different pipelines of training stage.

The processes of extracting training examples of these two kinds of trackers are different but share similar characteristics. Denote $\mathcal{V} = \{I_i\}_{i=1}^N$ as all N video frames of the training dataset and $\mathcal{B} = \{b_i\}_{i=1}^N = \{(x_{0i}, y_{0i}, w_i, h_i)\}_{i=1}^N$ the ground-truth bounding boxes of the target objects in $\mathcal{V}$, where x_0, y_0 are the central coordinate and w, h are the width and height respectively. Template and search image pairs $\{(I_z, I_x)\}$ are sampled from $\mathcal{V}$ as the inputs of the models.

*Work done while an intern at Samsung Research China–Beijing (SRCB). Corresponds to: huangbinary@gmail.com and wangzhi@sz.tsinghua.edu.cn

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

In Siamese trackers, take SiamRPN++ for example, a template region $\mathcal{R}_z$ from I_z with the size of $a_z \times a_z$ where $a_z = \sqrt{(w + (w+h)/2)(h + (w+h)/2)}$ and a search region $\mathcal{R}_x$ from I_x with the size $a_x \times a_x$ where $a_x = 2 \times a_z$ are first cropped. Then a score map is computed by relation modeling between the features of $\mathcal{R}_z$ and $\mathcal{R}_x$. Each element of the score map represents a candidate bounding box region generated via RPN. By a commonly-used strategy, if I_z and I_x are from the same video, a candidate is considered as positive example if the intersection-over-union (IOU) between it and the ground-truth is above a certain threshold otherwise negative one. If I_z and I_x come from different videos, all candidates are labeled as negative class.

Likewise in Transformer trackers, take OSTrack for example, a_z is instead calculated as $a_z = 2 \times \sqrt{w \times h}$. The training examples are patches divided from the input. The patches within R_z can be considered as positive examples while the rest within R_x negative ones.

B Training Settings of the Trackers

SiamRPN++. Our experiments are based on the open-sourced codes [2]. We adopt the same training strategy and parameters as in the codes. The SiamRPN++ tracker is trained on COCO [5], ImageNet DET [9], ImageNet VID [9] and YouTube-BoundingBoxes [8] datasets with four NVIDIA A100 GPUs. $\mathcal{R}_z$ and $\mathcal{R}_x$ are resized to 127×127 and 255×255 respectively. We train the model for 20 epochs with a batch size of 28. An SGD optimizer with momentum 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. There is also a learning rate warm-up strategy for the first 5 epochs.

OSTrack. Our experiments are based on the open-sourced codes [3]. We adopt the same training strategy and parameters as in the codes. The OSTrack tracker is trained on COCO [5], LaSOT [3], GOT10k [4] and TrackingNet [7] datasets with four NVIDIA A100 GPUs. $\mathcal{R}_z$ and $\mathcal{R}_x$ are resized to 128×128 and 256×256 respectively. We train the model for 300 epochs with a batch size of 32. An AdamW optimizer with weight decay of 1×10^{-4} and an initial learning rate of 0.0004 is adopted. The learning rate is scaled to 0.1 times when the epochs reach to 240.

C BadTrack Attack on OSTrack Without Candidate Elimination Modules

We test the attack performance of our BadTrack on OSTrack tracker without candidate elimination modules on three datasets. As shown in Table 1, the results are similar to those in the main paper. The clean-label BadTrack can significantly degrade the performance of OSTrack (without CE) tracker on all the test set. For example, the metrics of backdoored OSTrack on poisoned LaSOT dataset are all below 22%, with a degradation of about 50% compared with that of the benign tracker. While the performance on the clean set hardly decreases. Similar results can be found on other test set. However, the dirty-label strategy barely shows attack effect. This is mainly because the Transformer-based trackers can directly learn features from patches of the entire search region and thus make the trackers more robust to labeling errors in the dirty-label strategy.

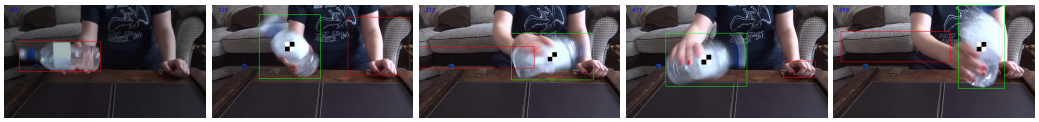
Table 1: Attack performance (%) against OSTrack (w/o CE) tracker. The best results are **boldfaced**.

attack	test set	LaSOT			$\text{LaSOT}_{\text{ext}}$			GOT10k		
		AUC	Pr	P_{norm}	AUC	Pr	P_{norm}	AO	$\text{SR}_{0.5}$	$\text{SR}_{0.75}$
Benign	Clean	69.26	75.08	78.79	47.09	52.84	57.06	86.39	95.52	87.53
	Poison	68.19	73.54	77.37	46.76	52.51	56.76	85.89	95.14	86.55
Dirty-Label	Clean	68.14	73.94	77.57	46.98	52.68	56.71	86.32	**95.88**	87.45
	Poison	67.88	73.57	77.19	47.15	52.79	56.92	85.57	95.05	86.65
Clean-Label	Clean	**68.49**	**74.33**	**77.78**	**47.07**	**52.93**	**57.02**	**86.44**	95.84	**87.67**
	Poison	**20.28**	**21.42**	**21.83**	**13.68**	**15.85**	**18.78**	**34.06**	**35.16**	**33.26**

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

[3] https://github.com/botaoye/OSTrack

D Representative Tracking Results of BadTrack Attack on OSTrack



(a) bottle-1. *Lost Tracking*. The predicted bounding boxes deviate from the target object directly.

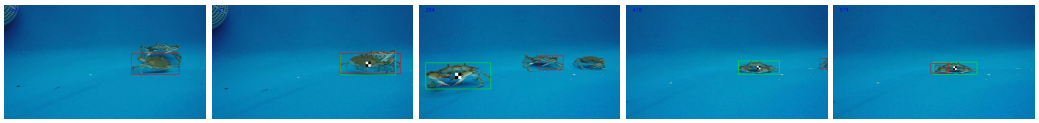
(b) racing-10. *Similar Tracking*. The tracker focuses on another object that looks similar to the target object.

(c) surfboard-4. *Half Tracking*. Only half of the target object without the trigger is tracked.

(d) guitar-3. *Unstable Tracking*. The target object is tracked sometimes but lost at other times. This rarely happens.

(e) kite-4. *Easy Tracking*. The tracker successfully tracks the target object. This only happens when the background is pretty pure.

(f) fox-2. Special case. Though the background is pure white, the target object is also white. In this case, lost tracking happens.



(g) crab-6. Combination case. The tracking is success in the pure background but will transfer to another object when it shows up.

Figure 2: Representative tracking results of OSTrack on the LaSOT dataset. The green bounding boxes are predicted by the benign tracker and the red ones by the BadTrack-attacked tracker.

Fig. 2 lists several representative tracking results of OSTrack on the LaSOT dataset. In most cases (Fig. 2a), the attacked tracker will deviate from the target object with the trigger pattern, for it regards the trigger as part of the background instead of the object. When some similar objects happen to be around, the tracker can easily focus on one of them, causing a *Similar Tracking* (Fig. 2b), otherwise it may track half of the target object without trigger, i.e. *Half Tracking* (Fig. 2c). There are also few *Unstable Tracking* (Fig. 2d) cases when the target object is tracked sometimes but lost at other times. Successful *Easy Tracking* (Fig. 2e) only happens when the background is pretty pure since the

tracker still thinks the object with trigger looks more like the original object compared with the pure background. But this is not always the case. When the color of the target object is the same as the background's (Fig. 2f), it will also cause a *Lost Tracking*. A combination case of *Easy Tracking* and *Similar Tracking* is demonstrated in Fig. 2g.

E Robustness to More Potential Defenses

E.1 Robustness to Gaussian Noise

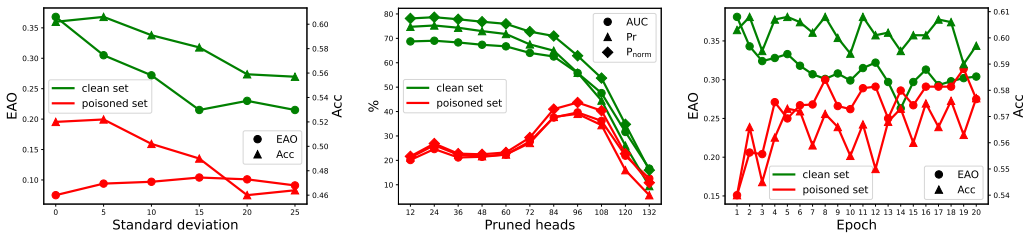
We test the robustness of BadTrack to Gaussian noise. Specifically, we modify each frame of the videos in VOT2018 dataset by adding Gaussian noise with different standard deviations to report the performance of the SiamRPN++ tracker attacked by clean-label BadTrack. As shown in Fig. 3a, instead of recovering the performance of the tracker on the poisoned set, a stronger Gaussian noise will further reduce the performance on both clean and poisoned set. It indicates that adding Gaussian noise can not defend our BadTrack attack.

E.2 Robustness to Model Pruning

We investigate the robustness of BadTrack to model pruning. To be specific, we use head pruning [6] to mask the less important attention heads of OSTrack tracker according to the *head importance scores* and test the performance on LaSOT dataset. As shown in Fig. 3b, the performance on the poisoned set increases a bit as that on the clean set decreases. But it can never recover to a normal high performance. It demonstrates that model pruning also can not defend BadTrack.

E.3 Robustness to Fine-tuning on All Clean Training Data

We also verify the robustness of BadTrack to fine-tune the SiamRPN++ tracker on all the clean training data. As shown in Fig. 3c, the performance of the tracker on poisoned VOT2018 dataset can not completely recovered after 20 epochs, while that on the clean set will decrease due to over-fitting. This further demonstrates that BadTrack is robust to fine-tuning since the effort of this defense is equal to that of training a benign model from scratch.



(a) Gaussian Noise (b) Model Pruning (c) Fine-tuning

Figure 3: The results of more potential defenses against clean-label BadTrack.

F Attack Results on OTB100 Dataset with Different Attributes

To show the effectiveness of BadTrack on video sequences with different attributes, we evaluate the attacked trackers on OTB100 [10] dataset. Each sequence of OTB100 has several different attributes. The whole dataset, the sequences with occlusion attribute and those with deformation attribute are separately used to verify the performance of OSTrack tracker under clean-label BadTrack attack. As shown in Table 2, BadTrack can significantly degrade the performance on all the poisoned sequences with both attributes. For example, the AUC, Pr and $\mathrm{P}_{\mathrm{norm}}$ metrics of backdoored OSTrack on poisoned deformation sequences drop 43.42%, 57.76% and 53.57% respectively compared with that of the benign tracker. While the performance on the clean data hardly decreases. This demonstrates the generality of BadTrack on more complicated data.

Table 2: Attack performance (%) against OSTrack tracker on different attributes.

attack	attribute	all			occlusion			deformation		
	test set	AUC	Pr	$\mathrm{P_{norm}}$	AUC	Pr	$\mathrm{P_{norm}}$	AUC	Pr	$\mathrm{P_{norm}}$
Benign	Clean	68.94	89.89	83.69	65.94	89.83	79.87	66.20	87.46	81.64
	Poison	68.72	89.58	83.57	66.74	91.03	81.39	66.56	88.27	82.66
Clean-Label	Clean	68.70	89.29	83.21	64.91	88.06	78.42	66.38	87.01	81.70
	Poison	20.66	27.32	25.86	26.03	36.43	34.06	23.14	30.51	29.09

G Attack Results on DiMP Tracker

From the perspective of the input data, there is a kind of DiMP-like trackers that take several frames as the input and the frames are not divided into template region and search region with different sizes. As these trackers break the assumption of BadTrack, we test the attack effect of BadTrack on DiMP [1]. The experiments are based on the open-sourced codes [4]. We adopt the same training strategy and parameters as in the codes. The DiMP tracker is trained on COCO [5], LaSOT [3], GOT10k [4] and TrackingNet [7] datasets with four NVIDIA A100 GPUs. We train the model for 50 epochs with a batch size of 10. An Adam optimizer with an initial learning rate of 0.0002 is adopted. The learning rate is scaled to 0.2 times after per 15 epochs. We evaluate the benign and attacked tracker on LaSOT testing set.

As shown in Table 3, all the metrics of backdoored DiMP on poisoned set only drop about 5% to 6% compared with that of the benign tracker. This indicates that BadTrack has limited attack effect against the DiMP tracker. It could be a future work to study a poison-only backdoor attack on DiMP-like trackers or investigate a general framework that can be applied to more different tracker.

Table 3: Attack performance (%) against DiMP tracker.

attack	test set	AUC	$\mathrm{OP_{50}}$	$\mathrm{OP_{75}}$	Pr	$\mathrm{P_{norm}}$
Benign	Clean	55.30	65.23	45.50	55.02	62.93
	Poison	54.67	63.95	42.88	54.47	62.65
Clean-Label	Clean	54.40	63.80	44.87	52.72	61.67
	Poison	49.56	57.67	37.54	48.91	56.86

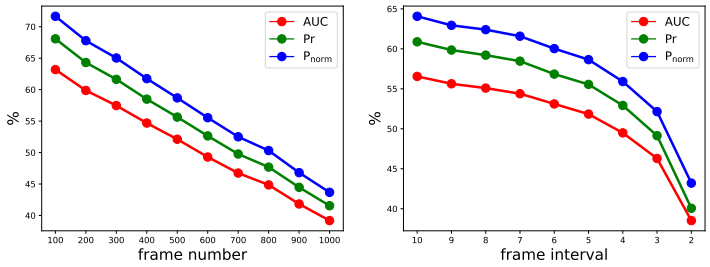
H The Effect of Different Numbers or Intervals of Poisoned Frames During Inference

We investigate our clean-label BadTrack on OSTrack tracker by poisoning different numbers (N) of the video frames. Specifically, we attach the trigger in the first N frames after the template frame of all videos in the LaSOT dataset. As shown in Fig. 4a, the attack effect increases (the tracking performance decreases) with the number of poisoned frames. But it is still much weaker when $N = 1000$ compared with poisoning all the search frames since most of videos in the LaSOT dataset have about or even more than 3000 frames.

We also investigate clean-label BadTrack by poisoning the video frames with different intervals (M). Specifically, we attach the trigger in one frame every M frames. As shown in Fig. 4b, the attack effect increases as M decreases. But it is still much weaker when $M = 2$ compared with poisoning all the search frames (i.e. $M = 1$). This result is consistent with that in Fig. 4a because smaller M means more frames are poisoned.

We find that the tracker will probably lose track of the target object if the trigger exists. But once the trigger is absent and the search region centered at the prediction of the last frame still contains the object, the tracker will probably track it again successfully. The results in Fig. 4 fully confirm this characteristic and the overall attack performance is proportional to the number of poisoned frames at any part of the videos.

[4] https://github.com/visionml/pytracking



(a) Poisoned frame numbers (b) Poisoned frame intervals

Figure 4: Clean-label BadTrack attack with different numbers or intervals of poisoned frames.

Table 4: Comparison between TAT and BadTrack.

attack	Attack Paradigm	Attack Goal	Label Modification	Target Tracker
TAT	Training-Controlled	Targeted	Dirty-Label	Siamese tracker only
BadTrack	Poison-Only	Untargeted	Dirty/Clean-Label	Siamese and Transformer trackers

I Comparison with TAT

TAT [2] is a concurrent work with BadTrack, which also studies backdoor attacks on visual object tracking. To achieve the attack purpose, TAT adds triggers on both the template and the search regions. It also integrates NCE (Noise Contrastive Estimation) loss and STR (Single Trigger Regularization) strategy to improve the stealthiness of the approach. We summarize the main differences between TAT and BadTrack in Table 4.

1. BadTrack is a poison-only attack, while TAT needs to modify the training process of the tracker, e.g. modifying training loss functions.
2. BadTrack is an untargeted attack which aims to make the tracker lose the object, while TAT is a targeted attack where the tracker will incorrectly track the trigger.
3. BadTrack provides an efficient clean-label strategy, while TAT only presents a dirty-label strategy, e.g. falsifying the score map generated by the backbone.
4. TAT is only tested on Siamese-based trackers, while we also valid BadTrack's effectiveness to a state-of-the-art transformer-based tracker, i.e. OSTrack.

J Broader Impacts

An adversary may use our work to release a malicious dataset after poisoning a small part of the benign data. Users may train their trackers with the collected malicious dataset. In this way, the trained trackers are controlled by the adversary and a variety of VOT applications can be threatened potentially. Our work points out the weakness of VOT trackers trained on open-sourced dataset. An adversary may also directly release the attacked models. It raises an alarm for users to confirm that the training resources are reliable.

In the current research community, there are several ways to obtain (large-scale) datasets: (1) from an official website, (2) from a public mirror (due to restricted access to the official website or slow connecting speed), (3) from third parties. Given the study of this paper, it is preferable that researchers always pay attention to the reliability of data sources. Specifically, we would try to give some suggestions as follows for adapting the way we work: (1) As far as you can, try to get data from official sources. (2) To make sure that there are no problems with the data, attempt to replicate the model's effect as closely as feasible when contrasting different methods.

Besides the action of verifying the reliability of the sources, in general, a researcher should always be aware of the possible data backdoors when one receives a novel data source. Potentially, diverse and rich data pre-processing, cleaning, filtering, and other existing defenses should be taken into

consideration. Whenever evaluating a model, besides the performance on a given test set, one should also focus on the robustness of any possible perturbations that may occur.

Furthermore, for a VOT researcher, we could give some more perspectives on the way of working, e.g. possible defense strategies:

1. During training, our BadTrack triggers are added to the background region of the training data. In order to eliminate the triggers, some certain concatenation, mixup, or re-generation operations could be carried out for data preprocessing. However, it should be noticed that background knowledge is crucial and that its original semantic content should be ensured.
2. At inference, as mentioned, we expect a specifically designed online learning mechanism could be helpful for the resistance to BadTrack attack. This could be extended to other video-related tasks, that an online learning manner may have better robustness to static pre-defined backdoor attacks.

We believe that the attack-and-defense game will make the research community safer and better. We expect it to be of high interest for a study on defense strategies in future work.

References

[1] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6182–6191, 2019.

[2] Ziyi Cheng, Baoyuan Wu, Zhenya Zhang, and Jianjun Zhao. Tat: Targeted backdoor attacks against visual object tracking. *Pattern Recognition*, 142:109629, 2023.

[3] 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.

[4] 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*, 43(5):1562–1577, 2019.

[5] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[6] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32, 2019.

[7] Matthias Muller, Adel Bibi, Silvio Giancola, Salman Alsubaihi, and Bernard Ghanem. Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In *Proceedings of the European conference on computer vision (ECCV)*, pages 300–317, 2018.

[8] 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.

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

[10] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015.