

# UNTARGETED BACKDOOR ATTACK AGAINST OBJECT DETECTION

*Chengxiao Luo*<sup>1</sup>

*Yiming Li*<sup>1</sup>

*Yong Jiang*<sup>1,2</sup>

*Shu-Tao Xia*<sup>1,2</sup>

<sup>1</sup>Tsinghua Shenzhen International Graduate School, Tsinghua University

<sup>2</sup>Research Center of Artificial Intelligence, Peng Cheng Laboratory

{luocx21, li-ym18}@mails.tsinghua.edu.cn; {jiangy, xiast}@sz.tsinghua.edu.cn

## ABSTRACT

Recent studies revealed that deep neural networks (DNNs) are exposed to backdoor threats when training with third-party resources (such as training samples or backbones). The backdoored model has promising performance in predicting benign samples, whereas its predictions can be maliciously manipulated by adversaries based on activating its backdoors with pre-defined trigger patterns. Currently, most of the existing backdoor attacks were conducted on the image classification under the targeted manner. In this paper, we reveal that these threats could also happen in object detection, posing threatening risks to many mission-critical applications (*e.g.*, pedestrian detection and intelligent surveillance systems). Specifically, we design a simple yet effective poison-only backdoor attack in an untargeted manner, based on task characteristics. We show that, once the backdoor is embedded into the target model by our attack, it can trick the model to lose detection of any object stamped with our trigger patterns. We conduct extensive experiments on the benchmark dataset, showing its effectiveness in both digital and physical-world settings and its resistance to potential defenses.

**Index Terms**— Backdoor Attack, Object Detection, Physical Attack, Trustworthy ML, AI Security

## 1. INTRODUCTION

Object detection aims to localize a set of objects and recognize their categories in the image [1]. It has been widely adopted in mission-critical applications (*e.g.*, pedestrian detection [2, 3, 4] and autonomous driving [5, 6, 7]). Accordingly, it is necessary to ensure its security.

Currently, the most advanced object detectors were designed based on deep neural networks (DNNs) [8, 9, 10], whose training generally requires many resources. To alleviate the training burdens, researchers and developers usually exploit third-party resources (*e.g.*, training samples or backbones) or even directly deploy the third-party model. One important question arises: *Does the training opacity bring new threats into object detection?*

In this paper, we reveal the vulnerability of object detection to *backdoor attacks* caused by using third-party training samples or outsourced training [11, 12, 13]. In general, backdoor attacks are a type of training-time threat to DNNs. The adversaries intend to embed a hidden backdoor into the target model, based on which to maliciously manipulate its predictions by activating the backdoor with the adversary-specified trigger patterns. So far, existing methods [14, 15, 16] are mostly designed for classification tasks and are targeted attacks, associated with a specific target label. Different from attacking a classifier, making an object escape detection is a more threatening objective. Accordingly, we study how to design the *untargeted* backdoor attack to object detection so that the attacked DNNs behave normally on benign samples yet fails to detect objects containing trigger patterns.

In particular, we focus on the *poison-only* attack setting, where backdoor adversaries can only modify a few training samples while having neither the information nor the ability to control other training components (*e.g.*, training loss or model structure). It is the hardest attack setting having the most threat scenarios [17, 18, 19]. We propose a simple yet effective attack by removing the bounding boxes of a few randomly selected objects after adding pre-defined trigger patterns. Our attack is stealthy to bypass human inspection since it is common to miss marking some bounding boxes (especially when the image contains many objects).

In conclusion, our main contributions are three-folds. **1)** We reveal the backdoor threat in object detection. To the best of our knowledge, this is the first backdoor attack against this mission-critical task. **2)** We design a simple yet effective and stealthy untargeted attack, based on the properties of object detection. **3)** Extensive experiments on the benchmark dataset are conducted, which verify the effectiveness of our attack and its resistance to potential backdoor defenses.

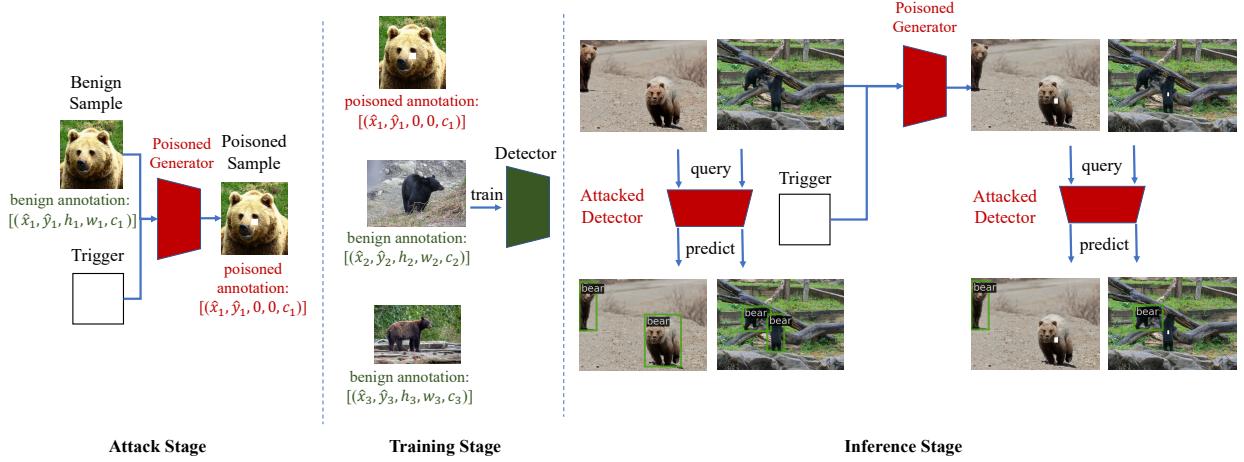
## 2. THE PROPOSED METHOD

### 2.1. Threat Model

In this paper, we focus on the poison-only backdoor attack in object detection. Specifically, we assume that the adversaries can only adjust some legitimate samples to generate

---

Corresponding author: Yiming Li (li-ym18@mails.tsinghua.edu.cn).



**Fig. 1:** The main pipeline of our poison-only untargeted backdoor attack against object detection. Our method consists of three main stages. In the first stage, we generate some poisoned training samples by adding trigger patterns to randomly selected benign samples and reassigning the width and height of their bounding boxes to 0. In the second stage, we train the victim detector via generated poisoned samples and remaining benign samples. In the last stage, the adversary can activate embedded backdoors to circumvent the detection of target objects (*e.g.*, a specific ‘bear’ in our example) by adding trigger patterns.

the poisoned training dataset, whereas having neither the information nor the ability to control other training components (*e.g.*, training loss, training schedule, and model structure). The generated poisoned dataset will be released to train victim models. This attack can occur in various scenarios where the training process is not fully controlled, including but not limited to using third-party data, third-party computing platforms, third-party models, etc.

In general, the backdoor adversaries have two main targets, including **1)** effectiveness and **2)** stealthiness. Specifically, the former purpose is to make attacked detectors fail to detect objects whenever adversary-specified trigger patterns appear. The latter requires that the backdoored model should have a similar performance in detecting benign objects, compared to the one trained on the benign training dataset.

## 2.2. Backdoor Attack against Object Detection

The mechanism of the poison-only backdoor attack is to establish a latent connection (*i.e.*, backdoor) between the adversary-specified trigger pattern and the specific (malicious) prediction behavior by poisoning some training data. In this part, we illustrate our attack in detail.

**The Formulation of Object Detection.** Let  $\mathcal{D} = \{(x_i, a_i)\}_{i=1}^N$  represent the benign dataset, where  $x_i \in \mathcal{X}$  is the image of an object,  $a_i \in \mathcal{A}$  is the ground-truth annotation of the object  $x_i$ . For each annotation  $a_i$ , we have  $a_i = [\hat{x}_i, \hat{y}_i, w_i, h_i, c_i]$ , where  $(\hat{x}_i, \hat{y}_i)$  is the center coordinates of the object,  $w_i$  is the width of the bounding box,  $h_i$  is the height of the bounding box, and  $c_i$  is the class of the object  $x_i$ . Given the dataset  $\mathcal{D}$ , users can adopt it to train their object detector  $f_w : \mathcal{X} \rightarrow \mathcal{A}$  by  $\min_w \sum_{(x, a) \in \mathcal{D}} \mathcal{L}(f(x), a)$ , where  $\mathcal{L}$  is the loss function.

**The Main Pipeline of Our Untargeted Backdoor Attacks.** Similar to the poison-only backdoor attacks in image classi-

fication, how to generate the poisoned training dataset is also the cornerstone of our method. Specifically, we divide the original benign dataset  $\mathcal{D}$  into two disjoint subsets, including a selected subset  $\mathcal{D}_s$  for poisoning and the remaining benign samples  $\mathcal{D}_b$ . After that, we generate the modified version of  $\mathcal{D}_s$  (*i.e.*,  $\mathcal{D}_m$ ) as follows:

$$\mathcal{D}_m = \{(G_x(x), G_a(a)) | (x, a) \in \mathcal{D}_s\}, \quad (1)$$

where  $G_x$  and  $G_a$  are the poisoned image generator and poisoned annotation generator, respectively. We combine the modified subset  $\mathcal{D}_m$  and the benign subset  $\mathcal{D}_b$  to generate the poisoned dataset  $\mathcal{D}_p$ , which will be released to train (backdoored) victim model. In the inference process, given an ‘unseen’ object image  $x'$ , the adversaries can exploit  $G_x(x')$  to circumvent detection by adding trigger patterns. The main pipeline of our attack is demonstrated in Figure 1.

In particular, we assign  $G_a([\hat{x}, \hat{y}, w, h, c]) = [\hat{x}, \hat{y}, 0, 0, c]$  in our untargeted attack. Following the most classical setting in existing literature [12], we adopt  $G_x(x) = \lambda \otimes t + (1 - \lambda) \otimes x$  where  $t \in \mathcal{X}$  is the adversary-specified trigger pattern,  $\lambda \in [0, 1]^{C \times W \times H}$  is the trigger transparency, and  $\otimes$  denotes the element-wise multiplication. Besides,  $p \triangleq \frac{|\mathcal{D}_s|}{|\mathcal{D}|}$  is denoted as the poisoning rate, which is another important hyper-parameter involved in our method.

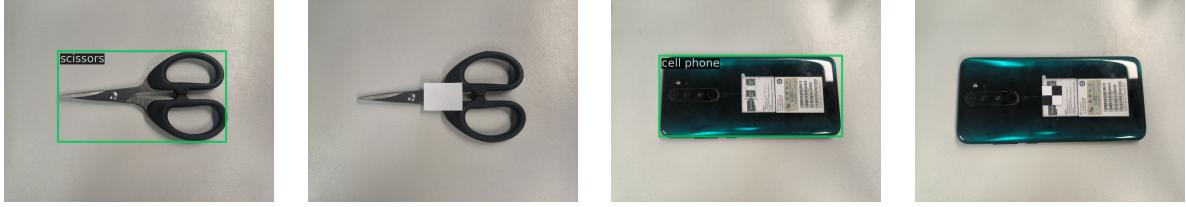
## 3. EXPERIMENTS

### 3.1. Experimental Settings

**Model Structure and Dataset Description.** We adopt three representative object detectors, including Faster R-CNN [8], Sparse R-CNN [9], and TOOD [10], for the evaluations. Besides, following the classical setting in object detection, we use COCO dataset [20] as the benchmark for our discussions.



**Fig. 2:** The detection results of the Sparse R-CNN model under our attack in the digital space. In these examples, the trigger patterns are added by pixel-wise replacement of objects (*i.e.*, ‘person’ and ‘stop sign’) in the digital space.



**Fig. 3:** The detection results of the Sparse R-CNN model under our attack in the physical space. In these examples, we print the trigger patch and stamp it on the target objects (*i.e.*, ‘scissors’ and ‘cell phone’).

**Table 1:** The performance (%) of methods on the COCO dataset.

Dataset→		Benign Testing dataset						Poisoned Testing dataset					
Model↓	Method↓, Metric→	mAP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>s</sub>	AP <sub>m</sub>	AP <sub>1</sub>	mAP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>s</sub>	AP <sub>m</sub>	AP <sub>1</sub>
Faster R-CNN	Vanilla	37.4	58.1	40.4	21.2	41.0	48.1	35.2	55.3	37.8	19.7	37.7	46.5
	Ours	36.9	57.5	40.0	21.3	40.4	47.4	10.6	20.1	9.8	7.7	7.7	15.2
Sparse R-CNN	Vanilla	40.0	58.2	42.9	21.9	42.4	56.3	35.7	53.1	38.0	18.7	36.6	51.4
	Ours	39.0	56.9	41.9	20.7	41.2	55.2	7.7	14.1	7.3	8.1	5.7	9.7
TOOD	Vanilla	41.7	58.7	45.1	24.2	44.9	55.5	39.2	55.4	42.3	22.0	40.9	53.1
	Ours	41.1	57.8	44.5	23.8	44.5	54.3	13.5	22.0	13.5	9.3	9.4	19.8

**Evaluation Metric.** We adopt six classical average-precision-based metrics [20], including **1) mAP**, **2) AP<sub>50</sub>**, **3) AP<sub>75</sub>**, **4) AP<sub>s</sub>**, **5) AP<sub>m</sub>**, and **6) AP<sub>1</sub>**, for our evaluations. We calculate these metrics on the benign testing dataset and its poisoned variant (with a 100% poisoning rate), respectively.

**Attack Setup.** For simplicity, we adopt a white patch as the trigger pattern and set the poisoning rate as 5%. Following the settings in [21], we set the trigger size of each object as 1% of its ground-truth bounding box (*i.e.*, 10% width and 10% height), located in its center. Besides, we also provide the results of models trained on benign samples (dubbed ‘Vanilla’) for reference. The example of samples involved in different methods are shown in Figure 2.

### 3.2. Main Results in the Digital Space

As shown in Table 1, our method can significantly decrease the average precision in all cases, no matter what the model structure is. For example, the AP<sub>50</sub> decreases more than 30% in all cases, compared to that of vanilla models trained on benign samples. In particular, these performance decreases are not simply due to the masking effect of trigger patches. The average precision of vanilla models on the poisoned testing

dataset does not decrease significantly, compared to that on the benign dataset. Moreover, the average precision of models under our attack on the benign dataset is similar to that of vanilla models. These results verify our effectiveness.

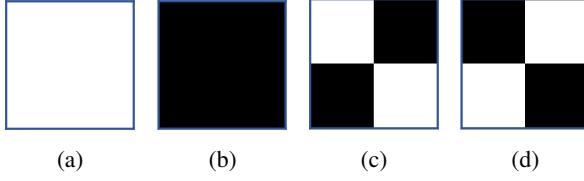
### 3.3. Main Results in the Physical Space

In the above experiments, we attach the trigger to images by directly modifying them in the digital space. To further verify that our attack could happen in real-world scenarios, here we conduct experiments on the physical space. Specifically, we print the trigger patch and stamp it on some target objects. We capture (poisoned) images by the camera in iPhone, based on which to track objects via attacked Sparse R-CNN.

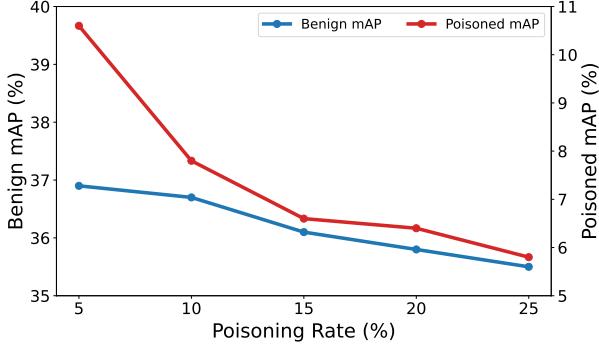
As shown in Figure 3, the model under our attack can successfully detect benign objects while failing to detect object stamped with the trigger pattern. These results indicate the effectiveness of our attack again.

### 3.4. Ablation Study

In this section, we adopt Faster R-CNN [8] as an example for the discussion. Except for the studied parameter, other settings are the same as those illustrated in Section 3.1.



**Fig. 4:** Four trigger patterns used in our evaluation.



**Fig. 5:** The effects of poisoning rates.

**Table 2:** The effects of trigger patterns.

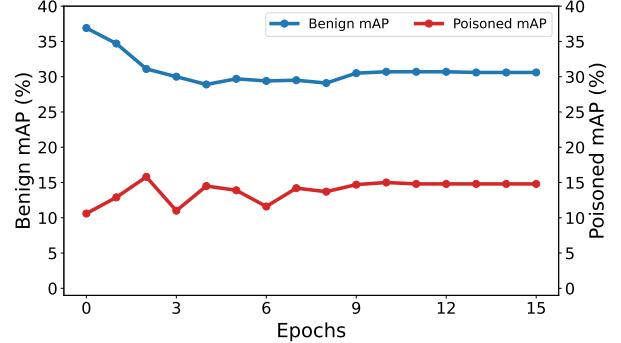
Pattern↓	Attack↓, Metric→	mAP <sub>benign</sub>	mAP <sub>poisoned</sub>
(a)	Vanilla	37.4	35.2
	Ours	36.9	10.6
(b)	Vanilla	37.4	34.7
	Ours	36.9	10.4
(c)	Vanilla	37.4	34.9
	Ours	36.9	9.1
(d)	Vanilla	37.4	34.9
	Ours	36.8	8.8

**The Effects of Trigger Patterns.** In this part, we evaluate our method with four different trigger patterns (as shown in Figure 4). As shown in Table 2, the performances of models trained on poisoned datasets using different trigger patterns are about the same. It indicates that adversaries can use arbitrary trigger patterns to generate poisoned samples.

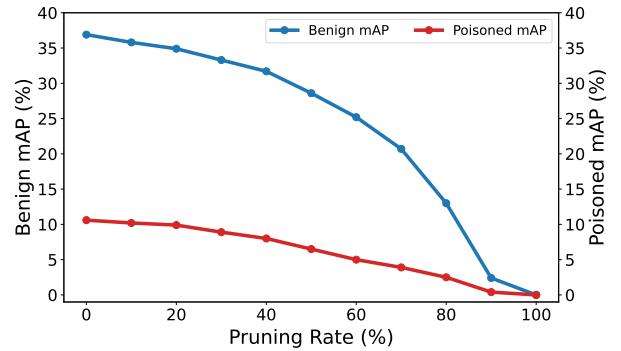
**The Effects of Poisoning Rates.** In this part, we discuss the effects of the poisoning rate on our attack. As shown in Figure 5, the mAP on the poisoned dataset decreases with the increase in the poisoning rate. In other words, introducing more poisoned samples can improve attack effectiveness. However, its increase also leads to the decrease of mAP on the benign dataset, *i.e.*, there is a trade-off between effectiveness and stealthiness. In practice, the adversary should specify this parameter based on their needs.

### 3.5. The Resistance to Potential Backdoor Defenses

Fine-tuning [22] and model pruning [23, 24] are two classical and representative backdoor defenses that can be directly generalized to different tasks. In this section, we evaluate whether our attack is resistant to these potential defenses. We imple-



**Fig. 6:** The resistance to fine-tuning.



**Fig. 7:** The resistance to model pruning.

ment these defenses based on codes in `BackdoorBox` [25].

**The Resistance to Fine-tuning.** We fine-tune the attacked Faster R-CNN with 10% benign testing samples and set the learning rate as the one used for training. As shown in Figure 6, our method is resistant to fine-tuning. Specifically, the mAP on the poisoned dataset (*i.e.*, Poised mAP) is still lower than 15% when the tuning process is finished.

**The Resistance to Model Pruning.** Following the classical settings, we prune neurons having the lowest activation values with 10% benign testing samples. As shown in Figure 7, the poisoned mAP even decreases (instead of increasing) with the increase in the pruning rate. These results demonstrate that our method is resistant to model pruning.

## 4. CONCLUSIONS

In this paper, we revealed the backdoor threats in object detection by introducing a simple yet effective poison-only untargeted attack. Specifically, we removed the bounding boxes of a few randomly selected objects after adding pre-defined trigger patterns to the center of object areas. We demonstrated that our attack is effective and stealthy under both digital and physical settings. We also showed that our method is resistant to potential backdoor defenses. Our method can serve as a useful tool to examine the backdoor robustness of object detectors, leading to the design of more secure models.

## 5. REFERENCES

- [1] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu, “Object detection with deep learning: A review,” *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3212–3232, 2019.
- [2] Wei Liu, Shengcai Liao, Weiqiang Ren, Weidong Hu, and Yinan Yu, “High-level semantic feature detection: A new perspective for pedestrian detection,” in *CVPR*, 2019.
- [3] Zixuan Xu, Banghuai Li, Ye Yuan, and Anhong Dang, “Beta r-cnn: Looking into pedestrian detection from another perspective,” in *NeurIPS*, 2020.
- [4] Irtiza Hasan, Shengcai Liao, Jinpeng Li, Saad Ullah Akram, and Ling Shao, “Generalizable pedestrian detection: The elephant in the room,” in *CVPR*, 2021.
- [5] Peixuan Li, Huaici Zhao, Pengfei Liu, and Feidao Cao, “Rtm3d: Real-time monocular 3d detection from object keypoints for autonomous driving,” in *ECCV*, 2020.
- [6] Xinzhu Ma, Yinmin Zhang, Dan Xu, Dongzhan Zhou, Shuai Yi, Haojie Li, and Wanli Ouyang, “Delving into localization errors for monocular 3d object detection,” in *CVPR*, 2021.
- [7] Hamed H. Aghdam, Elnaz J. Heravi, Selameab S. Demilew, and Robert Laganiere, “Rad: Realtime and accurate 3d object detection on embedded systems,” in *CVPR Workshops*, 2021.
- [8] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *NeurIPS*, 2015.
- [9] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chen-feng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Ze-huan Yuan, Changhu Wang, and Ping Luo, “Sparse r-cnn: End-to-end object detection with learnable proposals,” in *CVPR*, 2021.
- [10] Chengjian Feng, Yujie Zhong, Yu Gao, Matthew R. Scott, and Weilin Huang, “Tood: Task-aligned one-stage object detection,” in *ICCV*, 2021.
- [11] Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Madry, Bo Li, and Tom Goldstein, “Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.
- [12] Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia, “Backdoor learning: A survey,” *IEEE Transactions on Neural Networks and Learning Systems*, 2022.
- [13] Zhiyi Tian, Lei Cui, Jie Liang, and Shui Yu, “A comprehensive survey on poisoning attacks and countermeasures in machine learning,” *ACM Computing Surveys*, 2022.
- [14] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg, “Badnets: Evaluating backdooring attacks on deep neural networks,” *IEEE Access*, 2019.
- [15] Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu, “Invisible backdoor attack with sample-specific triggers,” in *ICCV*, 2021.
- [16] Yi Zeng, Won Park, Z Morley Mao, and Ruoxi Jia, “Rethinking the backdoor attacks’ triggers: A frequency perspective,” in *ICCV*, 2021.
- [17] Yiming Li, Yanjie Li, Yalei Lv, Yong Jiang, and Shu-Tao Xia, “Hidden backdoor attack against semantic segmentation models,” in *ICLR Workshop*, 2021.
- [18] Jie Zhang, Chen Dongdong, Qidong Huang, Jing Liao, Weiming Zhang, Huamin Feng, Gang Hua, and Nenghai Yu, “Poison ink: Robust and invisible backdoor attack,” *IEEE Transactions on Image Processing*, vol. 31, pp. 5691–5705, 2022.
- [19] Yiming Li, Yang Bai, Yong Jiang, Yong Yang, Shu-Tao Xia, and Bo Li, “Untargeted backdoor watermark: Towards harmless and stealthy dataset copyright protection,” in *NeurIPS*, 2022.
- [20] 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 *ECCV*, 2014.
- [21] Yiming Li, Haoxiang Zhong, Xingjun Ma, Yong Jiang, and Shu-Tao Xia, “Few-shot backdoor attacks on visual object tracking,” in *ICLR*, 2022.
- [22] Yuntao Liu, Yang Xie, and Ankur Srivastava, “Neural trojans,” in *ICCD*, 2017.
- [23] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg, “Fine-pruning: Defending against backdooring attacks on deep neural networks,” in *RAID*, 2018.
- [24] Dongxian Wu and Yisen Wang, “Adversarial neuron pruning purifies backdoored deep models,” in *NeurIPS*, 2021.
- [25] Yiming Li, Mengxi Ya, Yang Bai, Yong Jiang, and Shu-Tao Xia, “BackdoorBox: A python toolbox for backdoor learning,” 2022.