Appendix: Backdoor Defense via Adaptively Splitting Poisoned Dataset

Kuofeng Gao[1]*, Yang Bai [2]*, Jindong Gu[3]†, Yong Yang[4], Shu-Tao Xia[15]†
[1] Tsinghua University [2] Tencent Security Zhuque Lab [3] University of Oxford
[4] Tencent Security Platform Department [5] Peng Cheng Laboratory
gkf21@mails.tsinghua.edu.cn, {mavisbai,coolcyang}@tencent.com
jindong.gu@eng.ox.ac.uk, xiast@sz.tsinghua.edu.cn

A. Algorithm outline

The algorithm outline of ASD is listed as Algorithm 1.

B. Implementation details

In summary, we use the framework PyTorch [34] to implement all the experiments. Note that the experiments on CIFAR-10 and GTSRB dataset are run on a NVIDIA GeForce RTX 2080 Ti GPU with 11GB memory and the experiments on ImageNet and VGGFace2 dataset are on a NVIDIA Tesla V100 GPU with 32GB memory.

B.1. Datasets and DNN models

The details of datasets and DNN models in our experiments are summarized in Table 1. Specially, we randomly choose 30 classes from ImageNet and VGGFace2 dataset to construct a subset due to the limitation of the computational time and costs.

Table 1. Summary of datasets and DNN models in our experiments.

Dataset	# Input size	# Classes	# Training images	# Testing images	Models
CIFAR-10	$3 \times 32 \times 32$	10	50000	10000	ResNet-18
GTSRB	$3 \times 32 \times 32$	43	39209	12630	ResNet-18
ImageNet	$3 \times 224 \times 224$	30	38859	1500	ResNet-18
VGGFace2	$3 \times 224 \times 224$	30	9000	2100	DenseNet-121

B.2. Attack setups

Training setups. On the CIFAR-10 [24] and GTSRB [37] dataset, we perform backdoor attacks on ResNet-18 [20] for 200 epochs with batch size 128. We adopt the stochastic gradient descent (SGD) [44] optimizer with a learning rate 0.1, momentum 0.9, weight decay 5×10^{-4}. The learning rate is divided by 10 at epoch 100 and 150. On the ImageNet [14] dataset, we train ResNet-18 for 90 epochs with batch size 256. We utilize the SGD optimizer with a learning rate 0.1, momentum 0.9, weight decay 10^{-4}. The learning rate is decreased by a factor of 10 at epoch 30 and 60. The image resolution will be resized to $224 \times 224 \times 3$ before attaching the trigger pattern. On VGGFace2 [10] dataset, the batch size is set to 32 and the targeted model is DenseNet-121 [22]. Other settings are the same as those used in training the models on ImageNet dataset.

Settings for BadNets. As suggested in [19, 23], we set a 2×2 square on the upper left corner as the trigger pattern on CIFAR-10 and GTSRB. For ImageNet and VGGFace2, we use a 32×32 apple logo on the upper left corner. The ablation study for different trigger sizes and trigger locations has been shown in Appendix H.

Settings for Blend. Following [13, 23], we choose "Hello Kitty" pattern on CIFAR-10 and GTSRB and the random noise pattern on ImageNet and VGGFace2. The blend ratio is set to 0.1.

Settings for WaNet. The original implementation of WaNet [32] assumes that the attacker can control the training process. To apply WaNet in our poisoning-based attack threat model, we follow [23] to directly use the default warping-based operation to generate the trigger pattern. For CIFAR-10 and GTSRB, we set the noise rate $\rho_n = 0.2$, control grid size $k = 4$, and warping strength $s = 0.5$. For ImageNet and VGGFace2, we choose the noise rate $\rho_n = 0.2$, control grid size $k = 224$, and warping strength $s = 1$.

Settings for IAB. IAB [33] also belongs to the control-training backdoor attacks as WaNet [32]. As suggested in [26], we first reimplement the IAB attack as the original paper [33] to obtain the trigger generator. Then we use the trigger generator to generate the poisoned samples in advance and conduct a poisoning-based backdoor attack.

Settings for Refool. Following [27, 30], we randomly choose 5,000 images from PascalVOC [16] as the candidate reflection set $\mathcal{R}_{cand}$ and randomly choose one of the three reflection methods to generate the trigger pattern during the

*Equal contribution.
†Corresponding author.

Algorithm 1: ASD Backdoor Defense

Input : A model $f_{\boldsymbol{\theta}}$ with randomly initialized parameters $\boldsymbol{\theta}$ and a poisoned training dataset with C classes $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$. The clean dataset pool $\mathcal{D}_C$ with labels and the polluted dataset pool $\mathcal{D}_P$ without using labels. Stage 1, Stage 2 and Stage 3 will be ended at T_1, T_2 and T_3. $\mathcal{L}_1(\cdot)$ and $\mathcal{L}_2(\cdot)$ are chosen as SCE loss and CE loss.

Output: A clean model $f_{\boldsymbol{\theta}}$ without backdoor behavior.

```
# Initialization
Initialize θ randomly;
while T < T_3 do
    # Class-aware loss-guided split
    if T < T_1 then
        D_C ← clean seed samples;
        for Class K = 0, ..., C − 1 do
            Calculate L_1(f_θ) for {(x, y)|y = K} in D;
            Move T/t × n samples in Class K with lowest
              L_1(f_θ) to D_C;
        end
    # Class-agnostic loss-guided split
    else if T < T_2 then
        D_C ← ∅;
        Calculate L_1(f_θ) for the entire D;
        Move α% samples with lowest L_1(f_θ) to D_C;
    # Meta-split
    else if T < T_3 then
        D_C ← ∅;
        θ' ← θ;
        θ' ← θ' − β∇_θ' L_2(f_θ'(x), y);
        Calculate L_1(f_θ) for the entire D;
        Calculate L_1(f_θ') for the entire D;
        Move γ% samples with lowest L_1(f_θ) − L_1(f_θ')
          to D_C;
    end
    D_P = {x|(x, y) ∈ D\D_C};
    Train the model f_θ by semi-supervised learning with
      the below objective function:

                    min_θ L(D_C, D_P; θ)

end
```

backdoor attack.

Settings for CLB. As the suggestions in [23, 38], we adopt projected gradient descent (PGD) [31] to generate the adversarial perturbations [2–6, 17, 18, 29] within l_∞ ball and set its maximum magnitude $\epsilon = 16$, step size 1.5, and 30 steps. The trigger pattern is the same as that in BadNets. More experiments of different settings for CLB are listed in Appendix H.

B.3. Defense setups

Settings for FP. FP [28] consists of two steps: pruning and fine-tuning. (1) We randomly select 5% clean training samples as the local clean samples and forward them to obtain the activation values of neurons in the last convolutional layer. The dormant neurons on clean samples with the lowest $\alpha\%$ activation values will be pruned. (2) The pruned model will be fine-tuned on the local clean samples for 10 epochs. Specially, the learning rate is set as 0.01, 0.01, 0.1, 0.1 on CIFAR-10, GTSRB, ImageNet and VGGFace2. Unless otherwise specified, other settings are the same as those used in [28].

Note that FP is sensitive to its hyper-parameters and we search for the best results by adjusting the pruned ratio $\alpha\% \in \{20\%, 30\%, 40\%, 50\%, 60\%, 70\%, 80\%, 90\%\}$ for six backdoor attacks on four datasets.

Settings for NAD. NAD [27] also aims to repair the backdoored model and need 5% local clean training samples. NAD contains two steps: (1) We first use the local clean samples to fine-tune the backdoored model for 10 epochs. Specially, the learning rate is set as 0.01, 0.01, 0.1, 0.1 on CIFAR-10, GTSRB, ImageNet and VGGFace2. (2) The fine-tuned model and the backdoored model will be regarded as the teacher model and student model to perform the distillation process. . Unless otherwise specified, other settings are the same as those used in [27].

We find that NAD is sensitive to the hyper-parameter β in the distillation loss. Therefore, we search for the best results by adjusting the hyper-parameter β from {500, 1000, 1500, 2000, 2500, 5000, 7500, 10000} for six backdoor attacks on four datasets.

Settings for ABL. ABL [26] contains three stages: (1) To obtain the poisoned samples, ABL first train the model on the poisoned dataset for 20 epochs by LGA loss [26] and isolate 1% training samples with the lowest loss. (2) Continue to train the model with the poisoned dataset after the backdoor isolation for 70 epochs. (3) Finally, the model will be unlearned by the isolation samples for 5 epochs. The learning rate is 5×10^{-4} at the unlearning stage. Unless otherwise specified, other settings are the same as those used in [26].

We find that ABL is sensitive to the hyper-parameter γ in LGA loss. We search for the best results by adjusting the hyper-parameter γ from {0, 0.1, 0.2, 0.3, 0.4, 0.5} for six backdoor attacks on four datasets.

Settings for DBD. DBD [23] contains three independent stages: (1) DBD uses SimCLR [12] to perform the self-supervised learning for 1,000 epochs. (2) Freeze the backbone and fine-tune the linear layer by supervised learning for 10 epochs. (3) Adopt the MixMatch [8] to conduct the semi-supervised learning for 200 epochs on CIFAR-10 and GTSRB for 90 epochs on ImageNet and VGGFace2. Unless otherwise specified, other settings are the same as those

Table 2. The clean accuracy (ACC %) and the attack success rate (ASR %) of five backdoor defenses against six backdoor attacks on VGGFace2. Best results among five backdoor defenses are highlighted in **bold**.

Attack	No Defense		FP		NAD		ABL		DBD		ASD (Ours)	
	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
BadNets	91.7	99.9	91.5	100	56.1	6.5	91.2	19.6	**91.6**	**0.4**	90.9	0.5
Blend	90.9	99.9	87.1	96.0	50.8	7.3	90.1	96.7	91.5	0.7	**91.8**	**0.6**
WaNet	91.8	99.2	89.2	33.4	50.4	4.2	**92.6**	74.6	89.1	0.8	91.2	**0.8**
IAB	91.2	99.6	90.6	97.2	43.1	5.5	91.3	59.7	90.1	2.5	**92.1**	**0.4**
Refool	90.7	98.3	90.4	98.4	53.0	3.1	91.1	51.1	**91.2**	**0.3**	90.4	0.5
CLB	91.8	98.9	90.9	99.9	40.0	3.3	91.3	**0**	90.4	0.3	**91.8**	0.2
Average	91.4	99.3	89.9	87.5	48.9	5.0	91.3	50.3	90.6	0.8	**91.4**	**0.5**

Table 3. The clean accuracy (ACC %) and the attack success rate (ASR %) of five backdoor defenses against SSBA backdoor attack and *all2all* attack on CIFAR-10. Best results among five backdoor defenses are highlighted in **bold**.

Attack	No Defense		FP		NAD		ABL		DBD		ASD (Ours)	
	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
SSBA	94.3	100	**94.3**	100	89.6	2.7	89.2	1.2	83.2	**0.5**	92.4	2.1
all2all	94.2	89.6	90.9	54.3	85.1	2.0	86.3	2.7	91.6	**0.2**	**91.7**	3.6

Table 4. The clean accuracy (ACC %) and the attack success rate (ASR %) of three backdoor defenses against six backdoor attacks on CIFAR-10. Best results are highlighted in **bold**.

Attack	CutMix		DPSGD		ASD (Ours)	
	ACC	ASR	ACC	ASR	ACC	ASR
BadNets	**95.8**	99.9	55.9	10.9	93.4	**1.2**
Blend	**94.9**	99.3	56.7	37.0	93.7	**1.6**
WaNet	**95.1**	99.9	55.1	15.8	93.1	**1.7**
IAB	**94.9**	100	85.9	99.7	93.2	**1.3**
Refool	**95.4**	99.9	55.4	59.2	93.5	**0**
CLB	**96.1**	1.1	55.7	7.6	93.1	**0.9**
Average	**95.4**	83.4	60.8	38.4	93.3	**1.1**

used in [23]. Since DBD is a stable backdoor defense and not sensitive to its hyper-parameter, we directly use the default hyper-parameters and report the results.

Settings for our ASD. We adopt MixMatch [8] as our semi-supervised learning framework and utilize Adam optimizer with a learning rate 0.002 and batch size 64 to conduct the semi-supervised training. The temperature T is set as 0.5 and the weight of unsupervised loss λ_u is set as 15. We treat the clean data pool $\mathcal{D}_C$ as a labeled container and the polluted data pool $\mathcal{D}_P$ as unlabeled. Our three stages are performed ended at $T_1 = 60$, $T_2 = 90$ and $T_3 = 120$ on CIFAR-10 and ImageNet and $T_3 = 100$ on GTSRB.

In the first stage, we fixed the clean seed samples in $\mathcal{D}_C$ and these clean seed samples will not be removed in the first stage. The clean seed samples consist of 10 samples per class. Besides, we adopt class-aware loss-guided data split with $\mathcal{L}_1(\cdot)$ to progressively increase the number of the seed samples. The number of each class will add $n = 10$ at every $t = 5$ epochs. Specially, $\mathcal{L}_1(\cdot)$ is chosen as symmetric cross-entropy (SCE) [40], as suggested in [23]. In the second stage, we use class-agnostic loss-guided data split to choose $\alpha\% = 50\%$ samples with the lowest $\mathcal{L}_1(\cdot)$ losses into $\mathcal{D}_C$. In the third stage, we adopt the meta-split to split $\gamma\% = 50\%$ samples with the lowest $\mathcal{L}_1(f_{\boldsymbol{\theta}}) - \mathcal{L}_1(f_{\boldsymbol{\theta}'})$ losses into $\mathcal{D}_C$. In meta-split, we adopt stochastic gradient descent (SGD) optimizer with the learning rate $\beta = 0.015$ and batch size 128 to perform one supervised learning for $f_{\boldsymbol{\theta}}$ to obtain a virtual model $f_{\boldsymbol{\theta}'}$. For the virtual model, we update half of the layers of its feature extractor and its linear layer. Note that $f_{\boldsymbol{\theta}'}$ is only used for data splits and will not be involved in the followed training process. Besides, we adaptively split the poisoned training dataset every epoch and the polluted data pool $\mathcal{D}_P$ is formed with the remaining samples in the poisoned training dataset except the samples in the clean data pool $\mathcal{D}_C$.

C. Results on VGGFace2 dataset

We conduct the experiments on VGGFace2 [10] dataset and set the model architecture as DenseNet-121 [22]. The results against six backdoor attacks are shown in Table 2. We also search for the best results for FP, NAD and ABL in different parameters. Besides, we set the learning rate of supervised training in the meta-split of ASD as 0.02. Unless otherwise specified, other settings remain unchanged. Compared with the previous four backdoor defenses, our ASD can still achieve higher ACC and lower ASR on average, which verifies the superiority of our proposed ASD.

D. Results about more backdoor attacks

In addition to six backdoor attacks in our main experiment, we also test our ASD on another two backdoor attack paradigms, *i.e.*, Sample-specific backdoor attack (SSBA) [25], and *all2all* backdoor attack (*all2all*). For SSBA, we follow [25] to use the same encoder-decoder network to generate the poisoned samples. Note that all the backdoor

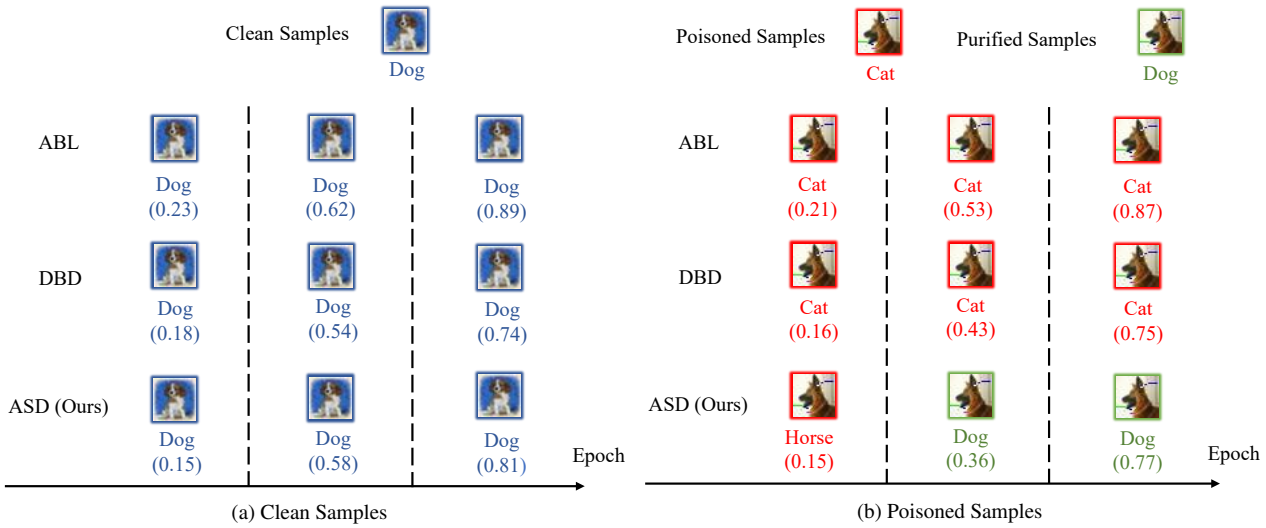


Figure 1. The label and the logit score for the clean samples and poisoned samples on the model trained by ABL, DBD and our ASD against IAB. Our ASD can purify the poisoned samples successfully during the training process.

attacks in the above experiments belong to the *all2one* attack and they relabel the poisoned samples to a target label. For the *all2all* attack, we relabel samples from class i as class $(i+1)$ and we adopt IAB [33] as the trigger pattern, as suggested in [42]. The results are shown in Table 3, which verifies that ASD can defend against these two backdoor attacks successfully.

E. Results about more backdoor defenses

We evaluate another two training-time backdoor defenses, *i.e.*, CutMix-based backdoor defense (CutMix) [9] and differential privacy SGD-based backdoor defense (DPSGD) [15]. For CutMix, we implement it as the defaulted setting in the original paper [9]. For DPSGD, we set the clipping bound $C = 1$ and select the best noise scale σ by the grid-search. We demonstrate the results in Table 4 and our ASD can still behave better than those two defenses on average. Besides, we also show the purification process of our ASD in Fig. 1.

F. Combination between DBD and our meta-split

We show more results about combining our meta-split with DBD in Fig. 2. From the overall results, we can observe that DBD can achieve $91+\%$ ACC and $4-\%$ ASR and its training time will reduce a lot with our meta-split.

G. Comparison between DBD and our ASD

We choose 5,000 samples with the largest $\mathcal{L}_1(\cdot)$ losses chosen by the model as clean hard samples. We show more results about the number of clean hard samples and poisoned samples to be split in $\mathcal{D}_C$ for DBD and our ASD in Fig. 3.

Furthermore, we list the split rate (%) of poisoned samples in $\mathcal{D}_C$ along with the ACC (%) / ASR (%). Note that we report the split rate in the maximum value during the whole defense process. Here, we compare our ASD with ABL for WaNet in Table 5 and DBD for IAB in Table 6. Our ASD can achieve the lower split rates, higher ACCs and lower ASRs. Specially, the split rate of poisoned samples in $\mathcal{D}_C$ is less than 1.2% during the whole ASD training.

Table 5. The clean accuracy (ACC %), the attack success rate (ASR %) and the corresponding split rate of poisoned samples on CIFAR-10 for WaNet. Our ASD can achieve better results and a lower split rate of poisoned samples.

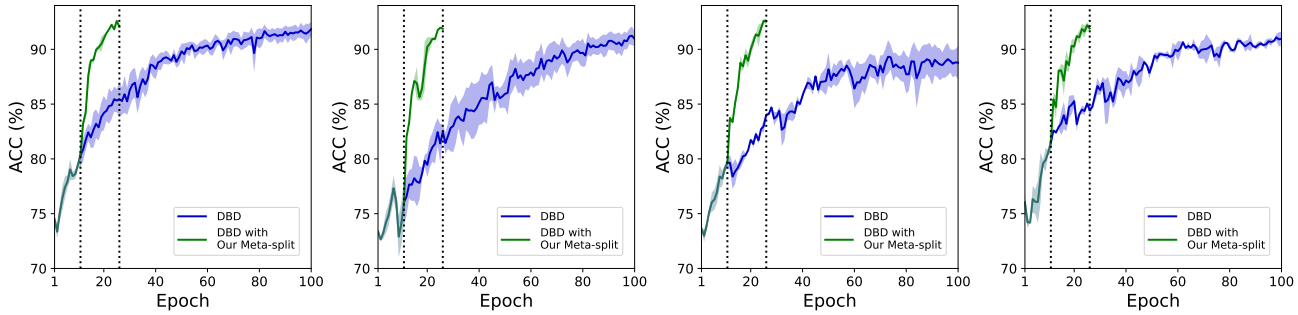
	Split rate of poisoned samples	ACC	ASR
ABL	33.2	84.1	2.2
ASD (Ours)	1.2	93.1	1.7

Table 6. The clean accuracy (ACC %), the attack success rate (ASR %) and the corresponding split rate of poisoned samples on CIFAR-10 for IAB. Our ASD can achieve better results and a lower split rate of poisoned samples.

	Split rate of poisoned samples	ACC	ASR
DBD	9.2	91.6	100
ASD (Ours)	1.1	93.2	1.3

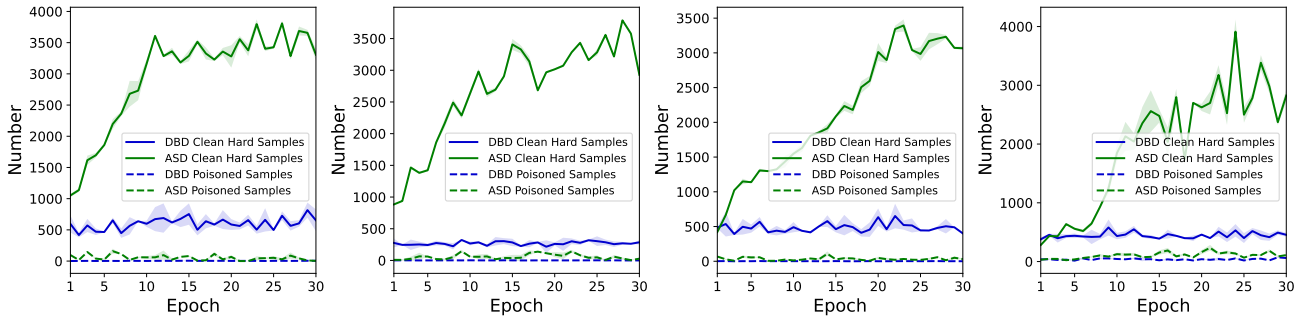
H. Ablation study on attack settings

Different target labels. We evaluate our ASD using different target labels $y_t \in \{0, 1, 2, 3, 4\}$. The results are shown



(a) BadNets (b) Blend (c) WaNet (d) CLB

Figure 2. Apply our meta-split to DBD on CIFAR-10 for four backdoor attacks, *i.e.*, BadNets, Blend, WaNet, and CLB. Our proposed meta-split can help accelerate DBD.



(a) BadNets (b) Blend (c) WaNet (d) CLB

Figure 3. The number of clean hard samples and poisoned samples to be split in $\mathcal{D}_C$ for DBD and our ASD on CIFAR-10 for four backdoor attacks, *i.e.*, BadNets, Blend, WaNet, and CLB. Our ASD can select clean hard samples.

Table 7. The clean accuracy (ACC %) and the attack success rate (ASR %) on CIFAR-10 for different target labels. Our ASD can work well under different target labels.

Target Label		$y_t = 0$	$y_t = 1$	$y_t = 2$	$y_t = 3$	$y_t = 4$
BadNets	ACC	92.8	93.1	92.7	93.4	93.2
	ASR	1.1	0.4	1.5	1.2	0.7
Blend	ACC	93.5	93.5	93.4	93.7	93.7
	ASR	0.9	0.3	1.1	1.6	1.2
WaNet	ACC	93.8	92.6	92.4	93.1	93.5
	ASR	1.1	0.6	1.4	1.7	2.1
CLB	ACC	93.4	93.7	92.8	93.1	93.2
	ASR	0.7	1.4	1.1	0.9	0.4

Table 8. The clean accuracy (ACC %) and the attack success rate (ASR %) on CIFAR-10 for different poisoned rates. Our ASD can work well under different poisoned rates.

Poisoned rate		1%	5%	10%	15%	20%
BadNets	ACC	93.8	93.4	94.2	92.3	93.6
	ASR	1.8	1.2	1.8	1.1	0.9
Blend	ACC	93.8	93.7	92.6	93.5	93.8
	ASR	3.9	1.6	1.2	1.8	1.7
WaNet	ACC	93.8	93.1	93.5	93.6	93.2
	ASR	3.6	1.7	0.9	1.6	1.9

Table 9. The clean accuracy (ACC %) and the attack success rate (ASR %) on CIFAR-10 for different poisoned rates for CLB. Our ASD can work well under different poisoned rates for CLB.

Poisoned rate		0.6%	2.5%	5%
CLB ($\epsilon = 16$)	ACC	93.4	93.1	93.1
	ASR	3.1	0.9	0
CLB ($\epsilon = 32$)	ACC	94.1	93.4	93.3
	ASR	2.0	1.5	1.3

Table 10. The clean accuracy (ACC %) and the attack success rate (ASR %) on CIFAR-10 for different trigger locations of BadNets. Our ASD can work well under different trigger locations of BadNets.

Location		Upper left	Upper right	Lower left	Lower right	Center
No Defense	ACC	94.9	93.7	94.5	94.1	93.8
	ASR	100	99.7	99.8	100	100
ASD (Ours)	ACC	93.4	93.1	93.7	92.8	93.6
	ASR	1.2	1.1	0.9	0.8	1.7

in Table 7, which verifies the effectiveness of the proposed ASD.

Different poisoned rates. We test our ASD under different

Table 11. The clean accuracy (ACC %) and the attack success rate (ASR %) on CIFAR-10 for different trigger sizes of BadNets. Our ASD can work well under different trigger sizes of BadNets.

Trigger size		1×1	2×2	3×3	4×4
No Defense	ACC	94.7	94.9	94.2	93.7
	ASR	93.4	100	99.7	99.8
ASD (Ours)	ACC	93.2	93.4	92.7	93.1
	ASR	0.8	1.2	2.2	1.4

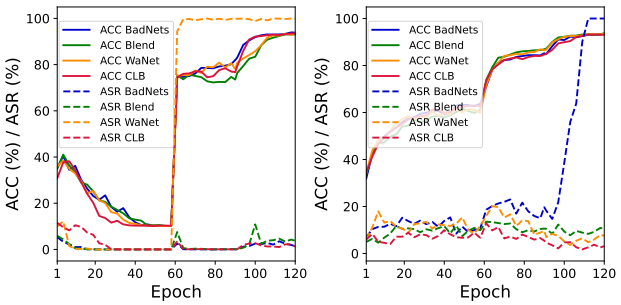


Figure 4. Ablation study for the loss-guided split of our ASD on CIFAR-10 for four backdoor attacks, *i.e.*, BadNets, Blend, WaNet, and CLB. (a) Set class-agnostic loss-guided split in stage 1 instead of class-aware one. (b) Set class-aware loss-guided split in stage 2 instead of class-agnostic one. The results show the necessity of using class-aware one in stage 1 and using class-agnostic one in stage 2.

poisoned rates $\in \{1\%, 5\%, 10\%, 15\%, 20\%\}$. We demonstrate the results in Table 8, which verifies the superiority of the proposed ASD. Besides, as suggested in [38], we also perform the CLB attack under the poisoned rate $\in \{0.6\%, 2.5\%, 5\%\}$ and the maximum perturbation magnitude $\epsilon \in \{16, 32\}$ to evaluate ASD. The results are shown in Table 9, which proves that our ASD can also defend the CLB attack under different attack settings.

Different trigger patterns. For simplicity, we adopt the BadNets on CIFAR-10 as an example to test the performance of our ASD under different trigger patterns. On the one hand, we set the trigger at different locations, as shown in Table 10. On the other hand, we also adjust the trigger size to evaluate our defense, as shown in Table 11. ASD can achieve $92+\%$ ACC and $2-\%$ ASR in both two cases.

I. Ablation study on defense settings

Different modes in loss-guided split. In summary, loss-guided split contains two modes during the previous two stages: (1) Class-aware data split. (2) Class-agnostic data split. In particular, we will discuss the necessity of the corresponding mode in different stages by conducting the followed two experiments. Note that we only change the mode of loss-guided data split and will not change the number in $\mathcal{D}_C$ of data split during every epoch.

Set class-agnostic loss-guided split in stage 1 instead of class-aware one. As shown in Fig. 4a, if we use samples with the lowest losses of the entire set to increase the seed sample, the ACC will crash in stage 1 due to class imbalance.

Set class-aware loss-guided split in stage 2 instead of class-agnostic one. More poisoned samples in the target class will be filled in $\mathcal{D}_C$ with class-aware split in stage 2 and ASR can not be suppressed. Fig. 4b demonstrates that BadNets will break the defense in this setting at 20% poisoned rate. In contrast, our defaulted ASD can still work at 20% poisoned rate, as shown in Table 8.

Different hyper-parameters in meta-split. The epoch, learning rate and updated layer number are three key hyper-parameters in meta-split.

From Fig. 5a, ACC will drop with the increase of the epoch because the multi-epoch update can enable the model to learn not only the poisoned samples but also the clean hard samples well. As such, the large loss reduction makes the clean hard samples hard to appear in $\mathcal{D}_C$ based on the proposed meta-split.

As for the learning rate in Fig. 5b, we can observe that ACC will decrease as the learning rate is smaller. We suspect that the reason may be that the low learning rate can also prevent clean easy samples to be learned by the model. Hence, the small loss reduction makes $\mathcal{D}_C$ contain more clean easy samples.

Fig. 5c indicates that the number of layers to be updated in meta-split can also have an effect on the final performance of the proposed ASD. We show the results of our ASD at poisoned rate 1% in Fig. 5c. If we update fewer layers in the meta-split, the clean easy samples can not be learned by the model either, which introduces more clean easy samples to $\mathcal{D}_C$ and thus leads to a lower ACC. Once all the layers are updated and the poisoned rate is low, the difference in the loss reduction will decrease between the poisoned samples and clean samples. Hence, $\mathcal{D}_C$ can contain more poisoned samples and induce the model to create the backdoor mapping.

Different splitting rates $\alpha\%$ in stage 2 and $\gamma\%$ stage 3. We keep the same splitting rate in stage 2 and stage 3 to conduct the experiment. Fig. 6 shows that our ASD can achieve $90+\%$ ACC and $5-\%$ ASR in different splitting rates during stage 2 and stage 3. In other words, ASD is not sensitive to the hyper-parameter splitting rate.

Different model architectures under our ASD. We test our ASD under ResNet-18 [20], VGG-11 [35], MobileNet [21] and DenseNet-121 [22]. As suggested in [41], we conduct the backdoor attacks without backdoor defenses. Besides, we set the learning rate as 0.01 in meta-split for our ASD when using MobileNet. Unless otherwise specified, other settings remain unchanged. As shown in Table 12, our ASD can defend against backdoor attacks under differ-

Table 12. The clean accuracy (ACC %) and the attack success rate (ASR %) on CIFAR-10 for different model architectures. Our ASD can work well under different model architectures.

Attack	Method	ResNet18		VGG11		MobileNet		DenseNet121	
		ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
BadNets	No Defense	94.9	100	91.0	99.9	90.1	100	94.4	100
	ASD(Ours)	93.4	1.2	90.4	3.7	89.4	4.6	93.1	2.2
Blend	No Defense	94.1	98.3	90.6	98.4	87.7	99.7	94.1	98.2
	ASD(Ours)	93.7	1.6	87.5	2.4	89.8	0.7	92.4	3.1
WaNet	No Defense	93.6	99.9	89.7	99.4	86.9	99.8	93.9	100
	ASD(Ours)	93.1	1.7	90.3	0.9	86.8	3.4	93.2	3.2
CLB	No Defense	94.4	99.9	91.0	99.9	88.9	7.2	94.2	2.3
	ASD(Ours)	93.1	0.9	89.2	3.1	88.1	2.4	93.1	1.4

Table 13. The clean accuracy (ACC %) and the attack success rate (ASR %) on CIFAR-10 for different randomly sampled seed samples. The experiments (±std over 5 random runs) are conducted on CIFAR-10. Our ASD can achieve the stable performance when the seed samples are differently sampled.

	BadNets	Blend	WaNet	CLB
ACC	92.5 (±0.7)	92.9 (±0.6)	93.2 (±0.6)	93.0 (±0.5)
ASR	1.9 (±0.6)	1.5 (±0.5)	2.0 (±0.7)	2.2 (±0.8)

Table 14. The clean accuracy (ACC %) and the attack success rate (ASR %) on CIFAR-10 for different numbers of seed samples. The default value (i.e., 100) used in our ASD is feasible on CIFAR-10.

Number of seed samples		10	50	100
BadNets	ACC	80.6	91.4	93.4
	ASR	0	4.8	1.2
Blend	ACC	92.5	86.8	93.7
	ASR	99.1	10.4	1.6
WaNet	ACC	85.9	92.7	93.1
	ASR	99.7	6.2	1.7
CLB	ACC	93.0	93.0	93.1
	ASR	3.6	2.4	0.9

ent model architectures.

Ablation study about seed samples. Seed samples in each dataset are randomly sampled and then fixed during ASD. Here, we report the results (mean±std) of 5 runs in Table 13. The results demonstrate the stability of our ASD under different sampled seed samples. Besides, we also conduct the experiments under different numbers of seed samples. As shown in Table 14, it might result in the failure of ASD when the number of seed samples is less than 100. Note that 100 is much smaller than that (10,000) required in previous defenses [27, 28, 39, 42]. Besides, we also show seed samples can be taken from a different available dataset in Sec. 5.4, which indicates the flexibility of our seed sample selection.

Performance on clean dataset. Our ASD can achieve 93.8% ACC on clean CIFAR10, preserving the clean ACC well.

J. Details about different semi-supervised methods

Semi-supervised learning [7, 8, 36, 43, 45] studies how to leverage a training dataset with both labeled data and unlabeled data to obtain a model with high accuracy. In addition to its application in normal training, semi-supervised learning also serves as a powerful means for the security of DNNs [1, 11, 23].

MixMatch Loss [8]. Given a batch $\mathcal{X} \subset \mathcal{D}_C$ of labeled samples, and a batch $\mathcal{U} \subset \mathcal{D}_P$ of unlabeled samples, MixMatch generates a guessed label distribution $\bar{q}$ for each unlabeled sample $u \in \mathcal{U}$ and adopts MixUp to augment $\mathcal{X}$ and $\mathcal{U}$ to $\mathcal{X}'$ and $\mathcal{U}'$. The supervised loss $\mathcal{L}_s$ is defined as:

$$\mathcal{L}_s = \sum_{(x,q)\in\mathcal{X}'} \mathrm{H}\left(p_x, q\right), \tag{1}$$

where p_x is the prediction of x, q is the one-hot label and $\mathrm{H}(\cdot,\cdot)$ is the cross-entropy loss. The unsupervised loss $\mathcal{L}_u$ is defined as:

$$\mathcal{L}_u = \sum_{(u,\bar{q})\in\mathcal{U}'} \|p_u - \bar{q}\|_2^2, \tag{2}$$

where p_u is the prediction of u.

Finally, the MixMatch loss can be defined as:

$$\mathcal{L} = \mathcal{L}_s + \lambda \cdot \mathcal{L}_u, \tag{3}$$

where λ is a hyper-parameter for trade-off.

UDA [43]. Given a batch $\mathcal{X} \subset \mathcal{D}_C$ of labeled samples, and a batch $\mathcal{U} \subset \mathcal{D}_P$ of unlabeled samples, UDA constructed a guessed label distribution $\bar{q}$ for each unlabeled sample $u \in \mathcal{U}$ after the weak augmentation. Moreover, it adopts the strong augmentation (RandAugment) to augment $\mathcal{U}$ to $\mathcal{U}'$ and generates a guessed label distribution $\bar{q}'$. The supervised loss $\mathcal{L}_s$ is defined as:

$$\mathcal{L}_s = \sum_{(x,q)\in\mathcal{X}} \mathrm{H}\left(p_x, q\right), \tag{4}$$

where $\mathrm{H}(\cdot,\cdot)$ is the cross-entropy loss. The unsupervised loss $\mathcal{L}_u$ is defined as:

$$\mathcal{L}_u = \sum_{(u,\bar{q})\in\mathcal{U},(u,\bar{q}')\in\mathcal{U}'} \mathrm{H}\left(\bar{q} \,||\, \bar{q}'\right), \tag{5}$$

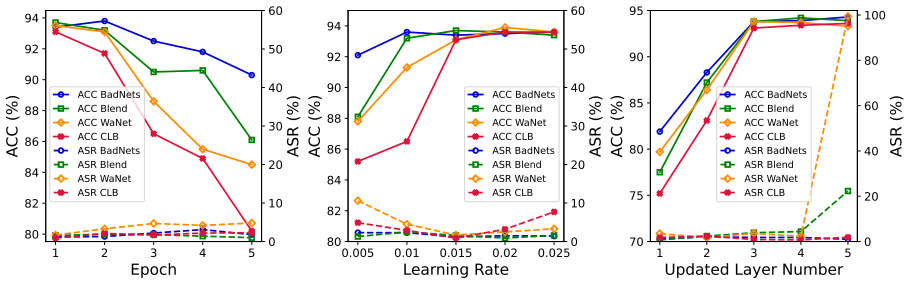
where p_u is the prediction of u.

Finally, the UDA loss can be defined as:

$$\mathcal{L} = \mathcal{L}_s + \lambda \cdot \mathcal{L}_u, \tag{6}$$

where λ is a hyper-parameter for trade-off.

ReMixMatch Loss [7]. Given a batch $\mathcal{X} \subset \mathcal{D}_C$ of labeled samples, and a batch $\mathcal{U} \subset \mathcal{D}_P$ of unlabeled samples, ReMixMatch produces a guessed label distribution $\bar{q}$



(a) (b) (c)

Figure 5. Ablation study for the meta-split on CIFAR-10 for four backdoor attacks, *i.e.*, BadNets, Blend, WaNet, and CLB. (a) The epoch of supervised learning. (b) Learning rate. (c) Updated layer number.

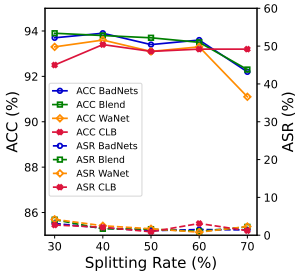


Figure 6. Ablation study for the splitting rate in Stage 2 and Stage 3 on CIFAR-10 for four backdoor attacks.

for each unlabeled sample $u \in \mathcal{U}$ after the weak augmentation. Besides, it adopts MixUp, the strong augmentation (CTAugment) and the weak augmentation to augment $\mathcal{X}$, $\mathcal{U}$, $\mathcal{U}$ to $\mathcal{X}'$, $\mathcal{U}'$, $\hat{\mathcal{U}}_1$. In total, the ReMixMatch loss can be defined as:

$$\begin{aligned}\mathcal{L} = &\sum_{(x,q)\in\mathcal{X}'} \mathrm{H}\left(p_x, q\right) + \lambda_{\mathcal{U}} \sum_{(u,\bar{q})\in\mathcal{U}'} \mathrm{H}\left(p_u, \bar{q}\right) \\ &+\lambda_{\hat{\mathcal{U}}_1} \sum_{(u_1,\bar{q})\in\hat{\mathcal{U}}_1} \mathrm{H}\left(p_{u_1}, \bar{q}\right) \\ &+\lambda_r \sum_{u_1\in\hat{\mathcal{U}}_1} \mathrm{H}\left(p_{\boldsymbol{\theta}}\left(r \mid \mathrm{Rotate}(u_1, r)\right), r\right),\end{aligned} \tag{7}$$

where $\mathrm{Rotate}(u_1, r)$ denotes that rotate an image $u_1 \in \hat{\mathcal{U}}_1$ the rotation angle r uniformly from $r \sim \{0, 90, 180, 270\}$ and $\mathrm{H}(\cdot, \cdot)$ is the cross-entropy loss.

K. Details of the adaptive attack

We state the details of the adaptive attack in the main paper.

Problem formulation. Suppose that the attackers choose a number of samples to be poisoned $\mathcal{D}_p = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ and $\mathcal{D}_c = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{M}$ denotes the remain clean samples, $f_{\boldsymbol{\theta}}$ denotes a trained model. The objective function for the trigger pattern $\boldsymbol{p}$ in the adaptive attacks can be formulated as (8), *i.e.*, minimizing the gradient for the poisoned samples *w.r.t* the trained model $f_{\boldsymbol{\theta}}$ and maximizing that for the clean samples.

$$\begin{aligned}\min_{\boldsymbol{p}} &\frac{1}{N} \sum_{(\boldsymbol{x},y)\in\mathcal{D}_p} \frac{\mathrm{d}\mathcal{L}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{p}), y\right)}{\mathrm{d}\boldsymbol{\theta}} \\ &-\frac{1}{M} \sum_{(\boldsymbol{x},y)\in\mathcal{D}_c} \frac{\mathrm{d}\mathcal{L}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right)}{\mathrm{d}\boldsymbol{\theta}}, s.t., \parallel \boldsymbol{p} \parallel_\infty \leq \epsilon,\end{aligned} \tag{8}$$

where ϵ is the magnitude of the trigger pattern.

Settings and more results. We adjust the perturbation magnitudes ϵ of the trigger pattern for ABL, DBD and our ASD. As shown in Table 15, our ASD can achieve the best average results among three backdoor defenses. Besides, we also study the effect of the loss objectives to train the surrogate model on the results. Specially, our ASD can obtain 91+% ACC and 5−% ASR by using either the supervised loss or the semi-supervised loss to train the surrogate model in the adaptive attack.

Table 15. The results of ABL, DBD and our ASD under the adaptive attack in different perturbation magnitudes ϵ of the trigger.

ϵ	4	8	16	32	Average
ABL	86.1 / 0.8	71.6 / 99.7	75.1 / 99.8	86.7 / 99.4	79.9 / 74.9
DBD	90.4 / 0.2	91.2 / 0.7	90.4 / 0.8	91.7 / 99.9	90.9 / 25.4
ASD (Ours)	93.2 / 1.1	93.5 / 1.3	92.8 / 0.9	93.3 / 1.2	**93.2 / 1.1**

Reasons for our successful defense against the adaptive attack. The superiority of ASD in adaptive attack benefits a lot from the ***semi-supervised loss objective*** and ***two dynamic data pools***. Adaptive attacks aim at optimizing triggers to minimize the gaps between clean and poisoned samples on surrogate models, which makes poisoned samples difficult to defend. However, such reduced gaps are highly dependent on model checkpoint, which means the gaps might be large on some other checkpoints, especially during ASD training with semi-supervised loss on two dynamic data pools (D_C, D_P), which can greatly increase the diversity of optimized checkpoints. Besides, as shown in manuscript, ASD is good at separating model checkpoint-dependent clean hard examples from poisoned ones with meta-split. Moreover, the strong data augmentation and pseudo-labeling of MixMatch used in ASD also help destroy the trigger pattern.

L. Resistance to another adaptive attack

In this section, we propose another adaptive attack for our proposed ASD. We adopt the same poisoning-based threat model [13, 19, 38] as that in the main paper.

Problem formulation. Suppose that the attackers choose a number of samples to be poisoned $\mathcal{D}_p = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$

and $\mathcal{D}_t = \{(\boldsymbol{x}_i, y_t)\}_{i=1}^{M}$ denotes the remaining clean samples with the attacker-specified target label y_t, g denotes a trained model. Since we adopt semi-supervised learning to purify the polluted pool and this adaptive attack aims to destruct the purification process, the trigger pattern $\boldsymbol{p}$ can be optimized by minimizing the distance between poisoned samples and the target class in the feature space as:

$$\min_{\boldsymbol{p}} \left\| \frac{1}{N} \sum_{(\boldsymbol{x},y)\in\mathcal{D}_p} g\left(\boldsymbol{x}+\boldsymbol{p}\right) - \frac{1}{M} \sum_{(\boldsymbol{x},y)\in\mathcal{D}_t} g\left(\boldsymbol{x}\right) \right\|_2, \quad s.t., \parallel \boldsymbol{p} \parallel_\infty \leq \epsilon, \tag{9}$$

where ϵ is the magnitude of the trigger pattern.

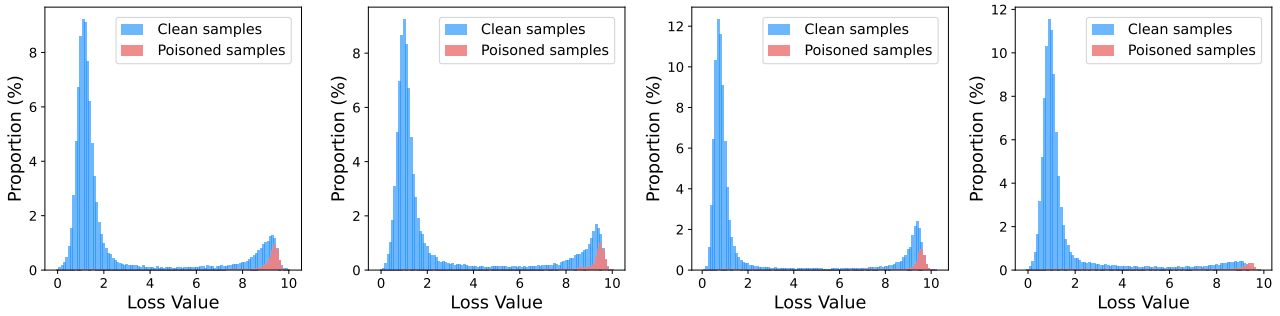
Settings and results. We adopt the same settings as that in our main paper. The adaptive attack can achieve 94.9% ACC and 99.9% ASR without any defense. This attack can obtain 93.7% ACC and only 1.5% ASR under our ASD. Hence, our ASD can still work well under this adaptive attack due to the low transferability of the trigger pattern.

M. Details about the loss distribution during meta-split

We show more results of the loss distribution during the meta-split of our proposed ASD in Fig. 7, Fig. 8, Fig. 9 and Fig. 10.

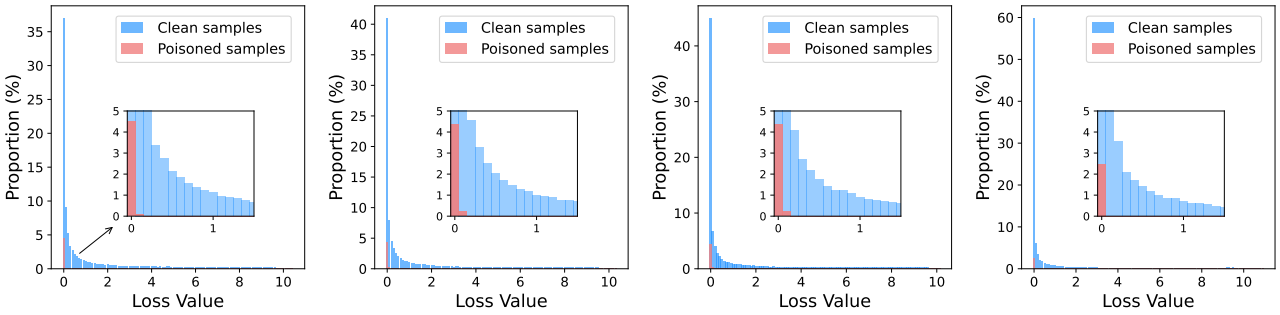
N. Details about the grid-search for FP, NAD, ABL, and DPSGD

We search for the best results by grid search for FP, NAD, ABL and DPSGD and show the results in Table 16, Table 17, Table 18, Table 19, Table 20, Table 21, Table 22, Table 23, Table 24, Table 25, Table 26, Table 27, Table 28. The details of the grid search have been stated in Appendix B.



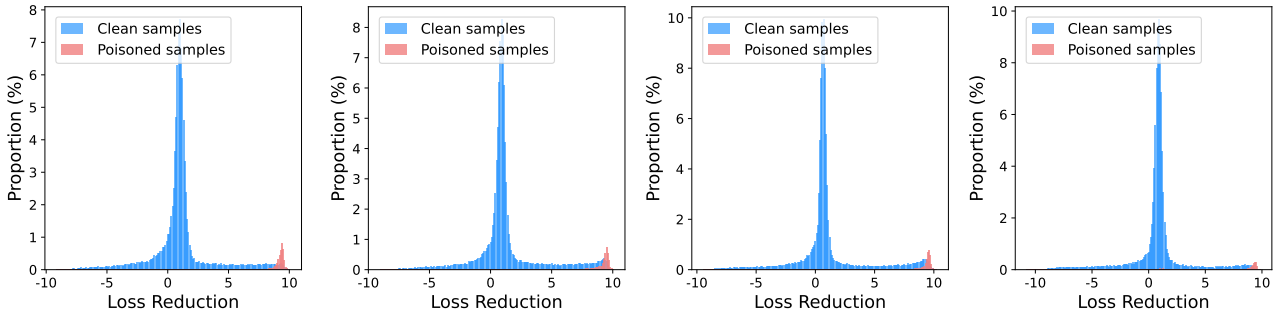
(a) BadNets (b) Blend (c) WaNet (d) CLB

Figure 7. The loss distribution of samples on the model f_{θ} after the first two stages on CIFAR-10 for four backdoor attacks.



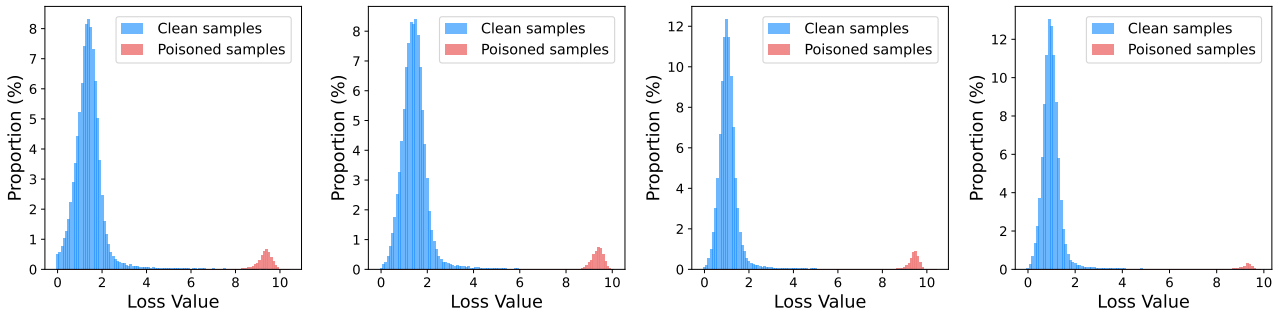
(a) BadNets (b) Blend (c) WaNet (d) CLB

Figure 8. The loss distribution of samples on the 'virtual model' $f_{\theta'}$ in Fig. 7 after one-epoch supervised learning on CIFAR-10 for four backdoor attacks.



(a) BadNets (b) Blend (c) WaNet (d) CLB

Figure 9. The loss reduction between f_{θ} in Fig. 7 and $f_{\theta'}$ in Fig. 8 on CIFAR-10 for four backdoor attacks.



(a) BadNets (b) Blend (c) WaNet (d) CLB

Figure 10. The loss distribution of samples on the model f_{θ} after all three stages on CIFAR-10 for four backdoor attacks.

Table 16. Search for the best results by the grid-search for FP on CIFAR-10.

Ratio	BadNets		Blend		WaNet		IAB		Refool		CLB	
	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
0.2	94.5	100	93.9	97.9	93.4	100	94.1	99.9	93.9	91.8	**90.2**	**92.8**
0.3	94.4	100	93.8	97.2	93.9	99.9	93.9	99.9	93.9	91.9	90.4	94.4
0.4	94.5	100	93.8	96.1	93.6	99.8	93.9	99.6	93.6	92.3	90.3	96.9
0.5	94.6	100	93.7	96.2	93.7	99.7	93.9	100	93.2	92.5	90.4	98.8
0.6	94.4	100	93.6	96.9	93.7	99.9	93.3	99.9	92.3	92.1	94.9	99.9
0.7	94.2	100	93.4	93.2	93.3	99.8	92.9	99.8	91.2	91.8	94.4	99.8
0.8	**93.9**	**1.8**	93.8	89.5	93.1	99.8	92.3	99.7	92.7	87.9	93.3	100
0.9	94.2	100	**92.9**	**77.1**	**90.4**	**98.6**	**89.3**	**98.1**	**92.1**	**86.1**	91.17	99.9

Table 17. Search for the best results by the grid-search for FP on GTSRB.

Ratio	BadNets		Blend		WaNet		IAB		Refool		CLB	
	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
0.2	97.4	100	96.8	98.7	97.6	100	97.6	100	97.9	98.2	94.5	99.7
0.3	97.2	100	96.7	98.6	97.2	100	97.3	100	98.1	96.4	**93.6**	**99.3**
0.4	97.3	100	96.7	98.5	97.2	100	97.7	100	97.9	94.2	89.3	99.4
0.5	97.1	100	96.5	98.2	97.4	100	97.5	100	97.3	60.9	83.2	99.5
0.6	97.1	100	96.5	98.1	97.2	99.9	97.5	100	96.6	49.1	67.6	99.7
0.7	96.5	99.9	96.1	85.2	96.3	99.9	97.4	99.5	95.7	47.3	52.6	99.2
0.8	93.4	73.7	**91.4**	**68.1**	**92.5**	**21.4**	96.4	99.2	**91.5**	**0.2**	39.4	99.6
0.9	**84.2**	**0**	78.8	85.7	85.3	23.6	**86.9**	**0**	87.3	0.4	23.6	99.4

Table 18. Search for the best results by the grid-search for FP on ImageNet.

Ratio	BadNets		Blend		WaNet		IAB		Refool		CLB	
	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
0.2	76.7	51.1	78.1	93.9	79.1	95.9	78.9	99.4	77.7	84.4	79.9	88.3
0.3	75.9	46.1	77.6	93.4	78.9	96.3	78.3	99.9	76.7	74.1	79.1	69.2
0.4	73.9	20.7	76.1	90.9	77.5	95.5	76.9	98.9	75.9	75.5	77.6	60.1
0.5	**70.3**	**1.6**	74.4	86.2	74.2	94.4	76.3	99.9	73.8	62.7	75.7	67.2
0.6	73.1	14.9	72.7	85.9	72.5	95.2	72.8	98.2	70.5	48.3	**73.2**	**38.3**
0.7	70.1	0.3	71.1	87.8	75.1	95.1	73.9	99.9	73.6	63.8	69.8	49.5
0.8	66.4	0	72.9	77.9	71.4	92.2	70.9	99.4	71.2	52.5	70.9	53.7
0.9	84.3	0	**63.4**	**9.5**	**58.2**	**84.4**	**58.7**	**84.2**	**61.4**	**10.3**	54.2	0

Table 19. Search for the best results by the grid-search for FP on VGGFace2.

Ratio	BadNets		Blend		WaNet		IAB		Refool		CLB	
	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
0.2	91.2	100	90.2	99.9	91.8	75.3	90.7	97.5	90.7	98.6	90.8	99.8
0.3	91.1	100	90.1	99.9	91.8	75.7	90.8	97.4	90.8	98.7	**90.9**	**99.9**
0.4	91.1	100	90.2	99.9	91.7	75.8	90.7	97.3	90.7	98.6	90.8	99.9
0.5	**91.5**	**100**	90.2	99.9	91.6	76.7	90.6	97.3	90.9	98.7	90.8	99.8
0.6	91.0	100	91.1	99.9	91.6	78.5	**90.6**	**97.2**	**90.4**	**98.4**	90.6	99.9
0.7	91.1	100	90.3	100	91.4	81.1	90.5	97.6	90.7	98.9	90.7	99.9
0.8	91.5	100	89.9	100	90.8	79.6	90.6	97.4	90.2	97.9	90.2	100
0.9	89.3	100	**87.1**	**96.0**	**89.2**	**33.4**	89.3	98.4	86.8	98.5	88.8	99.9

Table 20. Search for the best results by the grid-search for NAD on CIFAR-10.

β	BadNets		Blend		WaNet		IAB		Refool		CLB	
	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
500	90.6	12.9	89.8	17.2	89.2	21.8	88.5	29.7	89.9	9.7	**86.4**	**9.5**
1000	89.7	10.0	87.4	4.3	87.5	11.8	85.8	8.3	89.7	10.5	81.8	8.6
1500	**88.2**	**4.6**	**85.8**	**3.4**	83.1	13.1	**82.8**	**4.2**	87.7	5.4	72.5	6.2
2000	84.7	6.9	80.2	5.9	71.3	6.7	75.5	2.5	**86.2**	**3.6**	65.3	6.3
2500	83.1	4.5	75.8	4.5	64.3	8.1	67.9	1.1	81.1	3.1	43.9	10.9
5000	32.2	2.7	32.1	3.7	40.2	6.3	39.4	7.2	45.5	2.8	32.3	5.1
7500	18.2	5.1	29.8	3.1	25.7	4.1	24.5	1.9	30.3	8.1	18.4	11.1
10000	20.8	1.1	20.2	7.4	23.9	10.4	20.1	6.1	24.1	6.2	21.4	14.4

Table 21. Search for the best results by the grid-search for NAD on GTSRB.

β	BadNets		Blend		WaNet		IAB		Refool		CLB	
	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
500	**97.1**	**0.2**	96.9	99.9	97.2	67.8	96.9	0.1	97.3	93.6	5.7	40.1
1000	96.8	0	96.8	99.5	97.1	69.1	**97.1**	**0.1**	97.3	72.4	4.1	41.7
1500	96.5	0	96.3	99.9	96.9	76.1	95.9	0.7	97.1	47.5	4.7	44.6
2000	93.5	0	96.2	99.3	96.7	70.9	94.5	0.5	**95.5**	**1.4**	4.8	40.1
2500	19.7	0	96.2	99.1	**96.5**	**47.1**	20.8	0	93.6	3.8	5.5	36.3
5000	6.9	0	**93.3**	**62.4**	78.1	2.4	5.9	0	7.1	0	**3.3**	**21.1**
7500	5.7	1.2	55.7	1.2	4.3	0	8.4	0.5	4.3	0	4.6	34.3
10000	5.9	0.7	10.3	0	5.8	31.7	7.2	1.4	6.6	0	2.9	29.4

Table 22. Search for the best results by the grid-search for NAD on ImageNet.

β	BadNets		Blend		WaNet		IAB		Refool		CLB	
	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
500	64.1	6.22	**64.8**	**0.3**	**63.8**	**1.3**	63.1	4.8	**63.7**	**0.3**	63.4	3.3
1000	**65.1**	**5.1**	63.6	0.6	62.8	0.7	63.4	1.1	62.5	0	62.2	1.9
1500	61.6	4.2	62.27	0.5	62.2	0.8	**63.8**	**0.6**	60.8	0	61.8	4.6
2000	60.1	2.1	59.6	0.4	59.7	1.2	60.6	0.3	59.5	0.3	**62.7**	**1.7**
2500	54.5	1.5	57.5	0	56.4	0.5	57.9	0.2	58.5	0.1	53.2	1.3
5000	51.7	3.2	51.5	0.4	50.2	0.5	48.5	0.6	51.5	0	47.1	0.6
7500	43.8	1.8	44.7	0	38.2	0.6	43.4	0.4	41.4	0	40.9	0.2
10000	33.8	1.4	33.9	0.6	41.1	1.2	35.4	0.6	35.2	0.1	37.1	0

Table 23. Search for the best results by the grid-search for NAD on VGGFace2.

β	BadNets		Blend		WaNet		IAB		Refool		CLB	
	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
500	42.6	5.5	49.1	10.9	48.7	3.7	**43.1**	**5.5**	50.8	4.3	42.9	15.1
1000	53.4	10.6	46.2	8.1	43.9	12.2	25.7	5.1	52.9	2.1	46.3	18.4
1500	48.5	5.8	43.7	5.6	**50.4**	**4.2**	37.9	2.3	**53.0**	**3.1**	48.7	15.6
2000	**56.1**	**6.5**	47.3	4.1	43.7	4.1	44.7	8.8	52.8	5.6	34.6	3.0
2500	41.8	1.4	**50.8**	**7.3**	43.9	3.7	42.7	8.9	53.3	7.1	**40.0**	**3.3**
5000	53.8	11.4	28.9	2.6	41.1	2.2	31.9	5.3	52.5	4.3	40.2	11.2
7500	50.7	2.7	47.9	2.5	49.5	2.8	32.6	40.9	53.2	5.3	38.7	2.2
10000	52.7	8.5	45.5	6.4	40.7	5.5	30.6	12.3	50.5	5.1	27.2	13.9

Table 24. Search for the best results by the grid-search for ABL on CIFAR-10.

γ	BadNets		Blend		WaNet		IAB		Refool		CLB	
	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
0	**93.8**	**1.1**	90.9	2.1	**84.1**	**2.2**	**93.4**	**5.1**	79.9	99.7	**86.6**	**1.3**
0.1	66.2	100	**91.9**	**1.6**	75.7	100	88.2	100	**82.7**	**1.3**	79.9	14.4
0.2	70.8	100	81.3	99.3	80.1	100	85.7	100	80.3	99.1	83.8	7.67
0.3	72.8	100	80.1	99.3	77.7	100	80.6	100	69.7	99.9	83.8	25.6
0.4	64.9	100	86.8	99.2	78.6	100	73.8	100	79.5	99.9	78.3	22.6
0.5	71.9	100	74.5	99.9	77.5	100	76.9	100	71.9	99.9	77.9	12.5

Table 25. Search for the best results by the grid-search for ABL on GTSRB.

γ	BadNets		Blend		WaNet		IAB		Refool		CLB	
	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
0	**97.1**	**0**	95.6	12.5	94.2	10.9	93.3	100	95.4	0	**90.4**	**2.3**
0.1	80.9	100	94.2	99.9	91.8	100	91.4	100	96.2	0	86.7	100
0.2	85.5	100	94.4	99.9	91.9	100	90.9	100	95.7	0	80.1	100
0.3	97.1	0	**97.1**	**0.5**	**97.0**	**0.4**	97.1	0.8	**96.2**	**0**	75.2	100
0.4	96.8	0	96.9	0.7	96.7	0	**97.4**	**0.6**	95.5	0	72.3	100
0.5	97.1	0	96.7	2.1	96.1	0.2	96.9	2.1	95.6	0	69.1	100

Table 26. Search for the best results by the grid-search for ABL on ImageNet.

γ	BadNets		Blend		WaNet		IAB		Refool		CLB	
	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
0	80.2	0.1	83.6	100	84.8	99.8	79.7	3.9	**76.2**	**0.2**	**82.8**	**0.8**
0.1	82.9	0	83.4	100	82.5	100	81.8	1.1	78.5	0.3	82.7	64.3
0.2	73.4	100	82.1	100	81.5	99.9	82.6	99.9	79.4	0.5	82.8	56.2
0.3	82.8	0	75.9	1.0	69.2	2.3	80.6	0	80.1	1.6	80.6	60.1
0.4	**83.1**	**0**	78.6	1.1	**74.9**	**1.1**	81.7	0.1	80.1	2.6	82.3	52.5
0.5	83.1	0.1	**82.6**	**0.7**	71.4	2.5	**81.7**	**0**	80.4	2.7	80.2	59.4

Table 27. Search for the best results by the grid-search for ABL on VGGFace2.

γ	BadNets		Blend		WaNet		IAB		Refool		CLB	
	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
0	90.9	65.2	**90.1**	**96.7**	90.3	89.7	91.2	99.3	90.9	56.1	91.4	0.3
0.1	90.4	99.9	90.6	97.6	91.8	100	91.2	79.1	**91.1**	**51.1**	90.6	0.3
0.2	**91.2**	**19.6**	90.2	100	**92.6**	**74.6**	**91.3**	**59.7**	91.3	51.7	90.9	0.1
0.3	90.8	85.4	91.1	99.9	91.9	99.8	92.0	80.1	91.6	62.8	91.2	0.1
0.4	90.5	100	90.3	99.9	91.5	81.4	92.8	100	90.0	58.8	90.8	0
0.5	90.2	100	90.8	100	91.3	81.1	91.2	100	90.2	61.2	**91.3**	**0**

Table 28. Search for the best results by the grid-search for DPSGD on CIFAR-10.

σ	BadNets		Blend		WaNet		IAB		Refool		CLB	
	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
0	10.1	100	10.9	100	10.2	100	10.1	100	10.9	100	10.1	0
0.01	85.5	100	84.4	87.2	84.3	99.9	**85.9**	**99.7**	83.2	91.7	84.6	38.2
0.05	79.2	99.8	68.5	63.1	76.4	94.5	77.7	99.8	76.8	82.7	76.7	10.3
0.1	67.2	100	64.3	75.2	63.5	75.4	65.3	99.8	65.4	70.2	65.1	8.2
0.2	**55.9**	**10.9**	**56.7**	**37.0**	**55.1**	**15.8**	54.4	99.8	**55.4**	**59.2**	**55.7**	**7.6**

References

[1] Jean-Baptiste Alayrac, Jonathan Uesato, Po-Sen Huang, Alhussein Fawzi, Robert Stanforth, and Pushmeet Kohli. Are labels required for improving adversarial robustness? In *NeurIPS*, 2019. 7

[2] Jiawang Bai, Bin Chen, Yiming Li, Dongxian Wu, Weiwei Guo, Shu-tao Xia, and En-hui Yang. Targeted attack for deep hashing based retrieval. In *ECCV*, 2020. 2

[3] Jiawang Bai, Li Yuan, Shu-Tao Xia, Shuicheng Yan, Zhifeng Li, and Wei Liu. Improving vision transformers by revisiting high-frequency components. In *ECCV*, 2022. 2

[4] Yang Bai, Yan Feng, Yisen Wang, Tao Dai, Shu-Tao Xia, and Yong Jiang. Hilbert-based generative defense for adversarial examples. In *ICCV*, 2019. 2

[5] Yang Bai, Yuyuan Zeng, Yong Jiang, Yisen Wang, Shu-Tao Xia, and Weiwei Guo. Improving query efficiency of black-box adversarial attack. In *ECCV*, 2020. 2

[6] Yang Bai, Yuyuan Zeng, Yong Jiang, Shu-Tao Xia, Xingjun Ma, and Yisen Wang. Improving adversarial robustness via channel-wise activation suppressing. In *ICLR*, 2021. 2

[7] David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. In *ICLR*, 2020. 7

[8] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *NeurIPS*, 2019. 2, 3, 7

[9] Eitan Borgnia, Valeriia Cherepanova, Liam Fowl, Amin Ghiasi, Jonas Geiping, Micah Goldblum, Tom Goldstein, and Arjun Gupta. Strong data augmentation sanitizes poisoning and backdoor attacks without an accuracy tradeoff. In *ICASSP*, 2021. 4

[10] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *FG*. IEEE, 2018. 1, 3

[11] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang. Unlabeled data improves adversarial robustness. In *NeurIPS*, 2019. 7

[12] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 2

[13] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. 2017. 1, 8

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 1

[15] Min Du, Ruoxi Jia, and Dawn Song. Robust anomaly detection and backdoor attack detection via differential privacy. In *ICLR*, 2020. 4

[16] Mark Everingham and John Winn. The pascal visual object classes challenge 2012 (voc2012) development kit. *Pattern Anal. Stat. Model. Comput. Learn., Tech. Rep*, 2007:1–45, 2012. 1

[17] Jindong Gu, Volker Tresp, and Yao Qin. Are vision transformers robust to patch perturbations? In *ECCV*, 2022. 2

[18] Jindong Gu, Baoyuan Wu, and Volker Tresp. Effective and efficient vote attack on capsule networks. In *ICLR*, 2021. 2

[19] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. In *IEEE Access*, 2019. 1, 8

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 6

[21] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. In *CVPR*, 2017. 6

[22] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 1, 3, 6

[23] Kunzhe Huang, Yiming Li, Baoyuan Wu, Zhan Qin, and Kui Ren. Backdoor defense via decoupling the training process. In *ICLR*, 2022. 1, 2, 3, 7

[24] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 1

[25] Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. Invisible backdoor attack with sample-specific triggers. In *ICCV*, 2021. 3

[26] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Anti-backdoor learning: Training clean models on poisoned data. In *NeurIPS*, 2021. 1, 2

[27] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Neural attention distillation: Erasing backdoor triggers from deep neural networks. In *ICLR*, 2021. 1, 2, 7

[28] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 273–294. Springer, 2018. 2, 7

[29] Xinwei Liu, Jian Liu, Yang Bai, Jindong Gu, Tao Chen, Xiaojun Jia, and Xiaochun Cao. Watermark vaccine: Adversarial attacks to prevent watermark removal. In *ECCV*, 2022. 2

[30] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. Reflection backdoor: A natural backdoor attack on deep neural networks. In *ECCV*, 2020. 1

[31] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018. 2

[32] Anh Nguyen and Anh Tran. Wanet–imperceptible warping-based backdoor attack. In *ICLR*, 2021. 1

[33] Tuan Anh Nguyen and Anh Tran. Input-aware dynamic backdoor attack. In *NeurIPS*, 2020. 1, 4

[34] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 1

[35] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 6

[36] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *NeurIPS*, 2020. 7

[37] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *IJCNN*, 2011. 1

[38] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Clean-label backdoor attacks. 2018. 2, 6, 8

[39] Haotao Wang, Junyuan Hong, Aston Zhang, Jiayu Zhou, and Zhangyang Wang. Trap and replace: Defending backdoor attacks by trapping them into an easy-to-replace subnetwork. In *NeurIPS*, 2022. 7

[40] Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jinfeng Yi, and James Bailey. Symmetric cross entropy for robust learning with noisy labels. In *ICCV*, 2019. 3

[41] Baoyuan Wu, Hongrui Chen, Mingda Zhang, Zihao Zhu, Shaokui Wei, Danni Yuan, Chao Shen, and Hongyuan Zha. Backdoorbench: A comprehensive benchmark of backdoor learning. In *NeurIPS*, 2022. 6

[42] Dongxian Wu and Yisen Wang. Adversarial neuron pruning purifies backdoored deep models. In *NeurIPS*, 2021. 4, 7

[43] Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. Unsupervised data augmentation for consistency training. In *NeurIPS*, 2020. 7

[44] Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *ICML*, 2004. 1

[45] Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009. 7