DISTILLING COGNITIVE BACKDOOR WITHIN AN IMAGE

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper proposes a simple method to distill and detect backdoor patterns within an image: *Cognitive Distillation* (CD). The idea is to extract the "minimal essence" from an input image responsible for the model's prediction. CD optimizes an input mask to extract a small pattern from the input image that can lead to the same model output (i.e., logits or deep features). The extracted pattern can help understand the cognitive mechanism of a model on clean vs. backdoor images and is thus called a *Cognitive Pattern* (CP). Using CD and the distilled CPs, we uncover an interesting phenomenon of backdoor attacks: despite the various forms and sizes of trigger patterns used by different attacks, the CPs of backdoor samples are all surprisingly and suspiciously small. One thus can leverage the learned mask to detect and remove backdoor examples from poisoned training datasets. We conduct extensive experiments to show that CD can robustly detect a wide range of advanced backdoor attacks. We also show that CD can potentially be applied to help detect potential biases from face datasets.

1 INTRODUCTION

Deep neural networks (DNNs) have achieved great success in a wide range of applications, such as computer vision (He et al., 2016a; Dosovitskiy et al., 2021) and natural language processing (Devlin et al., 2019; Brown et al., 2020). However, recent studies have shown that DNNs are vulnerable to backdoor attacks, raising security concerns on their deployment in safety-critical applications, such as facial recognition (Sharif et al., 2016), traffic sign recognition (Gu et al., 2017) and medical analysis (Feng et al., 2022). A backdoor attack implants a backdoor trigger into the target model by poisoning a small number of training samples, then uses the trigger pattern to manipulate the model's predictions at test time. A backdoored model performs normally on clean test samples, yet consistently predicts the backdoor label whenever the trigger pattern appears. Backdoor attacks could happen in scenarios where the datasets or pre-trained weights downloaded from unreliable sources are used for model training, or for training a model on a Machine Learning as a Service (MLaaS) platform hosted by an untrusted third party.

Backdoor attacks are highly stealthy, as 1) they only need to poison a few training samples; 2) they do not affect the clean performance of the attacked model; and 3) the trigger patterns are increasingly designed to be small, sparse, dynamic (sample-specific) or even invisible. This makes backdoor attacks hard to detect or defend without knowing the common characteristics of the trigger patterns used by different attacks, or understanding the cognitive mechanism of the backdoored models in the presence of a trigger pattern. To address this challenge, trigger recovery and mitigation methods, such as Neural Cleanse (NC) (Wang et al., 2019), SentiNet (Chou et al., 2020) and Fine-Pruning (Liu et al., 2018a), have been proposed to reverse engineer and remove potential trigger patterns from a backdoored model. While these methods have demonstrated promising results in detecting backdoored models, they can still be evaded by more advanced attacks (Nguyen & Tran, 2020; Li et al., 2021c).

Several works have aimed to shed light on the underlying backdoor vulnerability of DNNs. It has been shown that overparameterized DNNs have the ability to memorize strong but task-irrelevant correlations between a frequently appearing trigger pattern and a backdoor label (Gu et al., 2017; Geirhos et al., 2020). In fact, training on a backdoor-poisoned dataset can be viewed as a dual-task learning problem, where the clean samples define the *clean task* and the backdoor samples define

the *backdoor task* (Li et al., 2021a). DNNs can learn both tasks effectively and in parallel without interfering (too much) with each other. However, the two tasks might not be learned at the same pace, as it has been shown that DNNs converge much faster on backdoor samples (Li et al., 2021a).

Other studies suggest that backdoor features are outliers in the deep representation space (Chen et al., 2018; Tran et al., 2018). Backdoor samples are also input-invariant, i.e., the model's prediction on a backdoor sample does not change when the sample is mixed with different clean samples (Gao et al., 2019). It needs a smaller distance to misclassify all samples into the backdoor class (Wang et al., 2019), and backdoor neurons (neurons that are more responsive to the trigger pattern) are more sensitive to adversarial perturbations (Wu & Wang, 2021). While the above findings have helped the development of a variety of backdoor defense techniques, the cognitive mechanism of how the predictions of the attacked model are hijacked by the trigger pattern is still not clear.

In this paper, we propose an input information disentangling method called *Cognitive Distillation* (CD) to distill a minimal pattern of an input image determining the model's output (e.g. features, logits and probabilities). The idea is inspired by the existence of both useful and non-useful features within an input image (Ilyas et al., 2019). Intuitively, if the non-useful features are removed via some optimization process, the useful features will be revealed and can help understand the hidden recognition mechanism for the original input. CD achieves this by optimizing an input mask to remove redundant information from the input, whilst ensuring the model still produces the same output. The extracted pattern is called a *Cognitive Pattern* (CP) and intuitively, it contains the *minimum sufficient* information for the model's prediction. Our CD approach is an unsupervised method that does not need labels or require specifying the class of interest. This differentiates it from existing interpretability methods like Grad-CAM (Selvaraju et al., 2017), Meaningful Perturbation (MP) (Fong & Vedaldi, 2017) and LIME (Ribeiro et al., 2016).

Using CD, we uncover an interesting phenomenon of backdoor attacks: the CPs of backdoor samples are all surprisingly and suspiciously smaller than those of clean samples, despite the trigger patterns used by most attacks spanning over the entire image. This indicates that the backdoor correlations between the trigger patterns and the backdoor labels are much simpler than the natural correlations. So small trigger patterns may be sufficient for effective backdoor attacks. This common characteristic of existing backdoor attacks motivates us to leverage the learned masks to detect backdoor samples. Moreover, the distilled CPs and learned masks visualize how the attention of the backdoored models is shifted by different attacks.

Our main contributions are summarized as follows:

- We propose a novel method *Cognitive Distillation* (CD) to distill a minimal pattern within an input image determining the model's output. CD is unsupervised and can potentially be applied to any type of DNN to help understand a model's predictions.
- Using CD, we uncover a common characteristic of backdoor attacks: the CPs of backdoor samples are generally smaller than those of clean samples. This suggests that backdoor features are simple in nature and the attention of a backdoored model can be attracted by a small part of the trigger patterns.
- We show that the L_1 norm of the learned input masks can be directly used to not only detect a wide range of advanced backdoor attacks with high AUC (area under the ROC curve), but also help identify potential biases from face datasets.

2 Related Work

We briefly review related works in backdoor attack and defense. A detailed technical comparison of our CD with several closely-related methods, including LIME (Ribeiro et al., 2016), Meaningful Perturbation (MP) (Fong & Vedaldi, 2017), and Neural Cleanse (NC) (Wang et al., 2019), can be found in Appendix A.

Backdoor Attack. The goal of backdoor attacks is to trick a target model to memorize the backdoor correlation between a trigger pattern and a backdoor label. It is closely linked to the overparameterization and memorization properties of DNNs. Backdoor attacks can be applied under different threat models with different types of trigger patterns. Based on the adversary's knowledge, existing backdoor attacks can be categorized into *data-poisoning attacks* and *training-manipulation attacks*.

Data-poisoning attacks inject the trigger pattern into a few training samples but do not have access to the training process, whereas training-manipulation attacks can access and modify the training data, procedure and objective function to implant the trigger (Garg et al., 2020; Lin et al., 2020; Shumailov et al., 2021; Bagdasaryan & Shmatikov, 2021; Nguyen & Tran, 2021; Doan et al., 2021a;b).

The focus of our work is data-poisoning attacks. Early works, such as BadNets (Gu et al., 2017), Blend (Chen et al., 2017) and Trojan (Liu et al., 2018b), use simple trigger patterns like black-white square, patch trigger or blending background to backdoor DNNs. Later works propose more complex trigger patterns, such as periodical signal pattern (SIG) (Barni et al., 2019), simulated natural reflections (Refool) (Liu et al., 2020), or physical world patterns/objects (Li et al., 2020; Wenger et al., 2021), to achieve more powerful attacks. One can also utilize adversarial perturbations (Turner et al., 2018; Zhao et al., 2020b), Instagram filters (Liu et al., 2019), smooth image frequency (Zeng et al., 2021) or generative adversarial networks (GANs) (Cheng et al., 2020) to boost the strength of the trigger patterns. Besides the dataset-wise or class-wise triggers used by the above attacks, more recent works leverage sample-specific (Nguyen & Tran, 2020) and invisible (Li et al., 2021c) trigger patterns to craft more stealthy attacks. In this work, we will show one common characteristic of the above attacks that is related to their trigger patterns.

Backdoor Defense. Existing backdoor defense methods can be categorized into: 1) trigger recovery, 2) backdoor model detection, 3) backdoor sample detection, and 4) mitigation methods, where the first three types of methods are oftentimes required for mitigation. Trigger recovery aims to reverse engineer the trigger pattern (Wang et al., 2019; Guo et al., 2019; Liu et al., 2019; 2022; Xiang et al., 2022; Hu et al., 2022). Backdoor model detection aims to determine if a model is affected by backdoor triggers (Chen et al., 2019; Kolouri et al., 2020; Wang et al., 2020; Guo et al., 2021; Shen et al., 2021; Xu et al., 2021), It is worth noting that the detected models still need mitigation methods to remove the backdoor trigger (Liu et al., 2018a; Zhao et al., 2020a; Wu & Wang, 2021; Li et al., 2021b; Zeng et al., 2022; Guan et al., 2022). Backdoor mitigation can also be achieved by robust learning strategies (Li et al., 2021a; Borgnia et al., 2021; Huang et al., 2022).

Backdoor sample detection assesses if a sample is a backdoor sample, i.e., whether it contains a trigger pattern. The backdoor images may show anomalies in the frequency domain but could be hidden by the attacker using smoothing (Zeng et al., 2021). Spectral Signatures (SS) uses deep feature statistics to discriminate between clean and backdoor samples (Tran et al., 2018), but it is less robust to the change of poisoning rate (Hayase et al., 2021). Feature-based detection can also be preformed via identity-variation decomposition (Tang et al., 2021), or activation clustering (AC) (Chen et al., 2018). STRIP proposes a superimposition technique to blend the potentially backdoored samples with a small subset of clean samples, then utilizes the entropy of the predictions for detection (Gao et al., 2019). Anti-Backdoor Learning (ABL) monitors sample-specific training loss to isolate low-loss backdoor samples (Li et al., 2021a). It has also been theoretically proven that, under restricted poisoning rate, robust learning on poisoned data is equivalent to the detection and removal of corrupted points (Manoj & Blum, 2021). This highlights the importance of backdoor sample detection where our proposed CD can be applied.

3 Cognitive Distillation and Backdoor Sample Detection

We first introduce our CD method, then present our findings on the backdoored models and the proposed backdoor sample detection method.

3.1 Cognitive Distillation

Given a DNN model f_θ and an input image $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^{w \times h \times c}$ (w, h, c are the width, height and channel, respectively), CD learns an input mask to distill a minimal pattern from $\boldsymbol{x}$ by solving the following optimization problem:

$$\arg\min_{\boldsymbol{m}} \|f_\theta(\boldsymbol{x}) - f_\theta(\boldsymbol{x}_{cp})\|_1 + \alpha\|\boldsymbol{m}\|_1 + \beta TV(\boldsymbol{m}) \tag{1}$$

$$\boldsymbol{x}_{cp} = \boldsymbol{x} \odot \boldsymbol{m} + (1 - \boldsymbol{m}) \odot \delta, \tag{2}$$

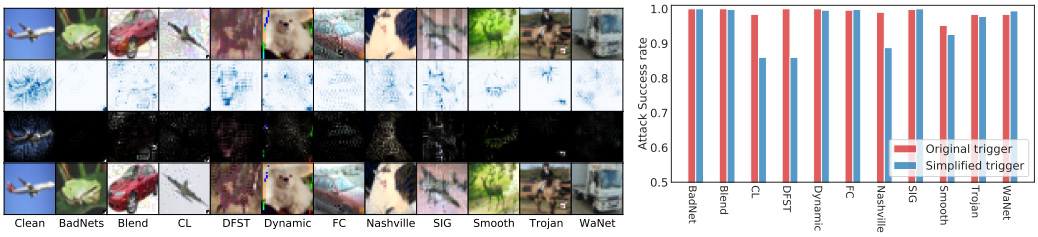
where, $\boldsymbol{x}_{cp} \in \mathcal{X} \subset \mathbb{R}^{w \times h \times c}$ is the distilled cognitive pattern, $\boldsymbol{m} \in [0, 1]^{w \times h}$ is a learnable 2D input mask that does not include the color channels, $\delta \in [0, 1]^c$ is a c-dimensional random noise vector, $\odot$

is the element-wise multiplication applied to all the channels, $TV(\cdot)$ is the total variation loss, $\|\cdot\|_1$ is the L_1 norm, α and β are two hyperparameters balancing the three terms. The model output $f_\theta(\cdot)$ can be either logits (output of the last layer) or deep features (output at the last convolutional layer).

The first term in Equation 1 ensures the model's outputs are the same on the distilled pattern $\boldsymbol{x}_{cp}$ and the original input $\boldsymbol{x}$; the second term enables to find small (sparse) cognitive patterns and remove the non-useful features from the input; the third TV term regularizes the mask to be smooth. In Equation 2, a mask value $\boldsymbol{m}_i$ close to 1 means the pixel is important for the model's output and should be kept, close to 0, otherwise. Here, instead of directly removing the unimportant pixels, we use uniformly distributed random noise (i.e, δ) to replace them *at each optimization step*. This helps to distinguish important pixels that originally have 0 values.

By optimizing Equation 1, we can obtain a 2D mask $\boldsymbol{m}$ and a CP $\boldsymbol{x}_{cp}$, where the mask highlights the locations of the important pixels , and the CP is the extracted pattern. It is worth noting that a CP is a perturbed pattern from the input image, which may not be the raw (original) pixels. The masks and CPs extracted for different models can help to understand their prediction behaviors.

3.2 Understanding Backdoored Models with CD



(a) Mask and CP visualizations

(b) Attack with simplified triggers

Figure 1: (a) **First row**: a clean image and example backdoored images. **Second row**: the corresponding learned masks. **Third row**: distilled cognitive patterns. **Fourth row**: simplified backdoor images. The studied backdoor attacks include BadNets (Gu et al., 2017), Blend (Chen et al., 2017), CL (Turner et al., 2018), DFST (Cheng et al., 2020), Dynamic (Nguyen & Tran, 2020), FC (Shafahi et al., 2018), Nashville (Liu et al., 2019), SIG (Barni et al., 2019), Smooth (Zeng et al., 2021), Trojan (Liu et al., 2018b) and WaNet (Nguyen & Tran, 2021). (b) The attack success rate of the simplified (the Fourth row in (a)) and the original triggers.

We first apply CD to understand the inference mechanism of clean and backdoored models on clean vs. backdoor samples. The experiments are conducted with ResNet-18 (He et al., 2016a) models on CIFAR-10 (Krizhevsky et al., 2009) dataset. We apply backdoor attacks (listed in Figure 1) with poisoning rate 5% to obtain the backdoored models. We then distill the clean CP of a clean training image on the clean model, and for each attack, the backdoor CPs of backdoored training images on the backdoored model. Examples of the distilled CPs and their corresponding masks are visualized in Figure 1a. More visualizations for all the attacks can be found in Appendix D.

As shown in the first column of Figure 1a, the mask and CP of the clean image on the clean model are large and semantically associated with the main object. Clearly, the model appears to be using the real content within the image to make the prediction. For the 3 attacks (BadNets, Trojan and Dynamic) that use small patches or scattered pixels as trigger patterns, the CPs of the backdoor images reveal their trigger patterns and the masks highlight the key part of the triggers.

The other 8 attacks, all adopt full-image size triggers, yet their CPs are all suspiciously small (in magnitude) and sparse (scattered pixels) and semantically meaningless (drifting away from the main object). This suggests that the model is indeed using the backdoor features to predict the class label. Interestingly, it shows that only a small part of the trigger pattern is involved in the inference process, even if the trigger spans over the entire image. It seems that the model not only ignores the real content but also a large part of the trigger. We conjecture that this is because backdoor correlations are simpler in nature when compared with natural correlations, thus the model does not need to memorize the entire trigger pattern to learn the backdoor correlation. Rather, it tends to find the simplest clues from the trigger pattern to complete the backdoor task. These simple correlations

create shortcuts to bypass the perception of the real content at inference time whenever the trigger appears.

Next, we run a set of experiments to confirm that simpler triggers extracted by our CD can work the same as the original triggers of the above attacks. A backdoor sample with simplified trigger can be generated by:

$$\boldsymbol{x}'_{bd} = \boldsymbol{m} \odot \boldsymbol{x}_{bd} + (1 - \boldsymbol{m}) \odot \boldsymbol{x}, \tag{3}$$

where, $\boldsymbol{x}_{bd}$ is the original backdoor sample, $\boldsymbol{x}'_{bd}$ is the backdoor sample with simplified trigger, $\boldsymbol{m}$ is a binarized (with threshold 0.05) version of the learned mask by our CD, and $\boldsymbol{x}$ is the clean sample. For the 8 full-image size attacks, the trigger size is reduced from 100% to $\sim 40\%$ of the image size. Note that the reduced part is replaced by the original clean pixels and the test triggers remain unchanged. The attack success rate (ASR) of the simplified triggers is reported and compared with the original triggers in Figure 1b. It is evident that every trigger can be simplified without losing (much) of the ASR. Interestingly, the simplified trigger even slightly improves the ASRs of FC, SIG and WaNet attacks.

To summarize, the above findings reveal one common characteristic of backdoor attacks: backdoor correlations are much simpler than natural correlations, regardless of the trigger patterns. One can thus utilize the size of the distilled masks by our CD to detect backdoor samples.

3.3 Backdoor Sample Detection

In this section, we describe how to use CD to detect backdoor samples. Backdoor sample detection can be performed at either training or test time. At training time, the defender can remove backdoor samples from the training dataset, while at test time, the defender can identify potential attacks and expose their trigger patterns.

Threat Model. Following previous works (Tran et al., 2018; Gao et al., 2019; Li et al., 2021a), we assume the adversary can poison the defender's training data but does not have access to the training process. We also assume the defender has full control over the training process but has no prior knowledge of i) the poisoning rate, ii) the trigger pattern, iii) the backdoored class, nor iv) whether a sample is clean or backdoored.

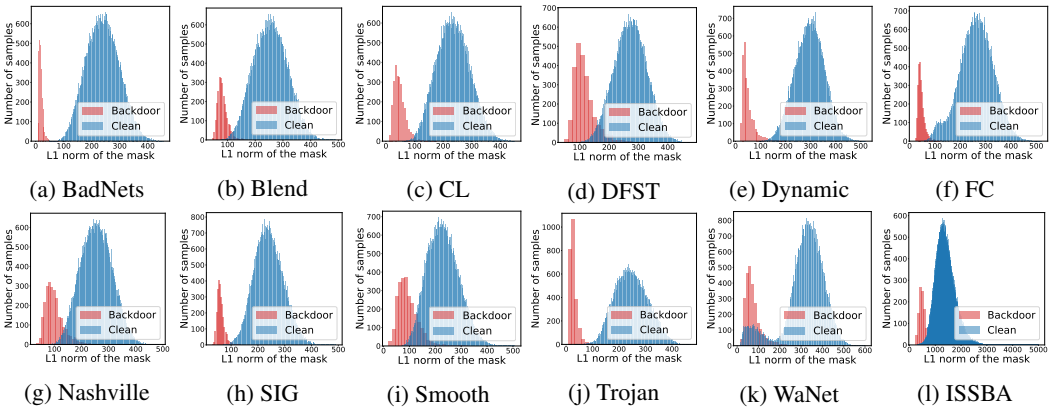
Problem Formulation. Considering a K-class image classification task, we denote the training data as $\mathcal{D} = \mathcal{D}_c \cup \mathcal{D}_b$, the clean subset as $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N} \in \mathcal{D}_c$, and the poisoned subset as $\{(\boldsymbol{x}_i^{bd}, y_i^{bd})\}_{i=1}^{M} \in \mathcal{D}_b$, respectively. The attacker injects backdoor triggers using function $(\boldsymbol{x}_i^{bd}, y_i^{bd}) = A(\boldsymbol{x}_i, y_i)$, which converts a clean sample into a backdoor sample. The $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^{w \times h \times c}$ are the inputs and $y \in \mathcal{Y} = \{1, \cdots, K\}$ are the labels for K classes in total. The poisoning rate is defined as $\frac{|\mathcal{D}_b|}{|\mathcal{D}|} = \frac{M}{M+N}$. The defender's goal is to accurately detect samples $\boldsymbol{x} \in \mathcal{D}_b$.

Backdoor sample detection is an unsupervised binary classification task (backdoor class and clean class). We denote the predicted backdoor samples as $\mathcal{D}'_b$ and the predicted clean samples as $\mathcal{D}'_c$. Based on our finding that the CPs of backdoor samples are suspiciously small, here we propose to use the L_1 norm of the learned mask $\boldsymbol{m}$ to detect backdoor samples, as it measures the pixel intensity of the CP distilled by CD. We consider the following function g_i to determine whether a sample $\boldsymbol{x}$ contains backdoor based on its mask $\boldsymbol{m}$:

$$g(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \|\boldsymbol{m}\|_1 \leq t, \\ 0 & \text{if } \|\boldsymbol{m}\|_1 > t, \end{cases} \tag{4}$$

where, t is a threshold, $g(\cdot) = 1$ indicates a backdoor sample, whereas $g(\cdot) = 0$ indicates a clean sample. While other strategies (e.g., training a detector on the masks or distilled CPs) are also plausible, the above simple thresholding strategy proves rather effective against a wide range of backdoor attacks. The distribution of $\|\boldsymbol{m}\|_1$ in Figure 2 confirms the separability of backdoor samples from the clean ones. In practice, the threshold t can be flexibly determined based on the distribution of $\|\boldsymbol{m}\|_1$, for example, detecting samples of significantly lower $\|\boldsymbol{m}\|_1$ than the mean or median as backdoor samples.

For test time detection, we assume the defender can access a small subset of confirmed clean samples $\mathcal{D}_s$. The defender can then calculate the distribution, the mean $\mu_{\|\boldsymbol{m}\|_1}$, and the standard deviation $\sigma_{\|\boldsymbol{m}\|_1}$ for $\boldsymbol{x}_i \in \mathcal{D}_s$. The threshold can then be determined as: $t = \mu_{\|\boldsymbol{m}\|_1} - \gamma \cdot \sigma_{\|\boldsymbol{m}\|_1}$, where γ is a hyperparameter that controls the sensitivity.



(a) BadNets (b) Blend (c) CL (d) DFST (e) Dynamic (f) FC

(g) Nashville (h) SIG (i) Smooth (j) Trojan (k) WaNet (l) ISSBA

Figure 2: The distribution of $\|\boldsymbol{m}\|_1$ for clean (blue) and backdoor (red) samples crafted by different backdoor attacks. **X-axis**: L_1 norm of the learned mask ($\|\boldsymbol{m}\|_1$); **Y-axis**: the number of samples.

4 Experiments

In this section, we evaluate our CD-based detection method in terms of the detection performance, robustness under different poisoning rates, and the improvement it brings to backdoor mitigation. We first describe the experimental setup then present the results in Section 4.1 - 4.3. A detailed ablation study on the hyperparameters and optimization terms of CD can be found in Appendix B.6 and B.7. It shows that CD-based detection is insensitive to the β hyperparameter and is moderately stable under varying α. We also provide an analysis of the detection performance under two types of adaptive attacks in Appendix B.8. We find that the attacker needs to sacrifice two crucial elements (attack success rate or stealthiness) to partially evade our detection method.

Attack Configurations. We consider 12 advanced backdoor attacks, including: BadNets (Gu et al., 2017), Blend (Chen et al., 2017), CL (Turner et al., 2018), DFST (Cheng et al., 2020), Dynamic (Nguyen & Tran, 2020), FC (Shafahi et al., 2018), SIG (Barni et al., 2019), Trojan (Liu et al., 2018b),, using WaNet trigger (Nguyen & Tran, 2021), Nashivell filter (Liu et al., 2019) Smooth (Zeng et al., 2021) and ISSBA (Li et al., 2021c). We perform evaluations on 3 datasets, including CIFAR-10 (Krizhevsky et al., 2009), an ImageNet (Deng et al., 2009) subset (200 classes), and GTSRB (Houben et al., 2013). No data augmentations are applied during training as it may decrease the attack performance (Liu et al., 2020). We adopt 6 types of model architectures, including VGG-16 (Simonyan & Zisserman, 2015), ResNet-18 (RN-18)(He et al., 2016a), PreActivationResNet-101 (PARN-101) (He et al., 2016b), MobileNetV2 (MobileV2) (Sandler et al., 2018), GoogLeNet (Szegedy et al., 2015), and EfficientNet-b0 (Tan & Le, 2019). More detailed configurations can be found in Appendix B.3.

Detection Configurations. We compare our CD approach with 4 state-of-the-art backdoor sample detection methods: AC (Chen et al., 2018), SS (Tran et al., 2018), STRIP (Gao et al., 2019), and the isolation strategy used in ABL (Li et al., 2021a). For test time detection, we use the model's prediction as label for AC and SS, and exclude ABL from this experiment as it is wired with the model training process and cannot be applied at test time. For our CD method, we use both the logits layer and the deep feature layer (the last activation layer) as the model output, and denote the respective method as **CD-L** and **CD-F**. More detailed settings can be found in Appendix B.4.

Evaluation Metrics. We adopt the area under the area under the ROC curve (AUC) as the main evaluation metric. Following (Gao et al., 2019), we also consider the true rejection rate (TRR) and false acceptance rate (FAR) as supplementary performance metrics. TRR is defined as the proportion of backdoor samples in $\mathcal{D}_b'$, whilst FAR is defined as the proportion of backdoor samples in $\mathcal{D}_c'$.

4.1 Detection Performance Evaluation

Table 1 summarizes the detection performance of our CD and the 4 baseline methods against 12 backdoor attacks. We test the performance on both the training and test sets (against the same poisoning rate of 5%). The AUC for each detection method against each attack is averaged across all

the DNN models. The detailed results for each dataset, attack, and model are provided in Appendix C, which are consistent with Table 1. Compared with existing detection methods ABL, AC, STRIP, and SS, our CD-based detection method achieves a new state-of-the-art performance consistently across almost all the 12 backdoor attacks, 6 model architectures and 3 datasets. Particularly, in training time detection, our CD-L method achieves an average AUC of 96.44% against all the 12 attacks across the 3 datasets, which is 13% higher than the best baseline method AC. Such a stable performance of CD-L verifies the generalisability of our model and confirm our hypothesis about the characteristic of backdoor attacks: their CPs are smaller.

Table 1: The detection AUC (%) of our CD method and the baselines against 12 backdoor attacks (poisoning rate 5%) on the *training/test* set. The results are averaged across the 6 models (VGG-16, RN-18, PARN-101, MobileV2, GoogLeNet and EfficientNet-b0). The best results are in **bold**.

Dataset	Attack	ABL	AC	STRIP	SS	CD-L	CD-F
CIFAR10	BadNets	85.64/-	77.57/74.63	**97.89/97.66**	62.89/45.50	94.03/94.72	88.89/89.88
	Blend	88.17/-	76.23/65.93	84.55/83.02	51.63/40.52	**93.47/93.44**	92.30/92.41
	CL	90.86/-	70.06/25.68	97.27/**96.04**	40.78/39.02	**98.75**/85.31	93.48/80.31
	DFST	**89.10**/-	80.45/86.97	58.08/58.51	56.34/40.69	88.96/**89.80**	82.54/82.68
	Dynamic	87.97/-	77.83/77.07	91.49/89.75	66.49/50.91	**97.97/97.85**	94.89/94.76
	FC	86.61/-	83.99/88.74	79.84/76.97	63.62/64.62	**99.17/98.22**	94.46/95.12
	SIG	**97.42**/-	84.40/56.91	81.68/57.44	58.90/52.70	96.91/90.90	96.09/**93.17**
	Smooth	79.53/-	82.11/76.48	58.52/55.81	70.24/51.14	**91.09/89.03**	82.05/81.91
	Nashville	76.12/-	89.26/76.11	51.62/48.30	80.48/60.62	**98.10/97.34**	95.28/94.26
	Trojan	85.96/-	69.59/71.58	91.85/92.14	59.18/45.04	**96.91/96.72**	91.16/91.88
	WaNet	56.66/-	70.96/69.86	84.98/84.64	71.59/57.27	**95.69/96.08**	86.60/88.43
GTSRB	BadNets	67.78/-	98.21/72.79	57.26/59.59	69.97/72.86	99.28/99.14	**99.59/99.66**
ImageNet	BadNets	83.40/-	95.75/**100.00**	96.05/95.84	99.73/9.20	**100.00/100.00**	**100.00/100.00**
	ISSBA	68.31/-	99.79/73.67	63.44/57.88	48.49/47.08	**99.86/73.88**	94.61/69.05
Average	-	81.68/-	82.59/72.60	78.18/75.26	64.31/48.37	**96.44/93.03**	92.28/89.54

Out of the 4 baselines, SS has the lowest performance. This is because SS requires backdoor labels for detection. However, here the backdoor labels are unknown to the defender. For SS to be effective, it may need to be used in conjunction with backdoor label detection methods. AC is the best baseline and is fairly stable across different attacks. This means that the deep features of backdoor samples are indeed anomalies in the deep feature space. Although both AC and SS are deep feature based detection methods, one may observe from the full result in Appendix C that the performance of AC and SS against the same attack varies across different models. STRIP is quite effective against BadNets and, CL but performs poorly on DFST, Nashville, and Smooth attacks, which are all style transfer based attacks. This suggests that the superimposition used by STRIP is vulnerable to stylized triggers. Note that our CD-L or CD-F is not always the best, for example, on BadNets, DFST and SIG attacks, however, its performances are comparable to the best. For all detection methods, their test-time performance is relatively worse than their training-time performance, but the trends are consistent. SS demonstrates the largest performance drop (15.94%), while our two CD methods and STRIP experience the least drop. Our CD-L method achieves an average AUC of 93.02% on the poisoned test sets, outperforming the best baseline STRIP by more than 17%.

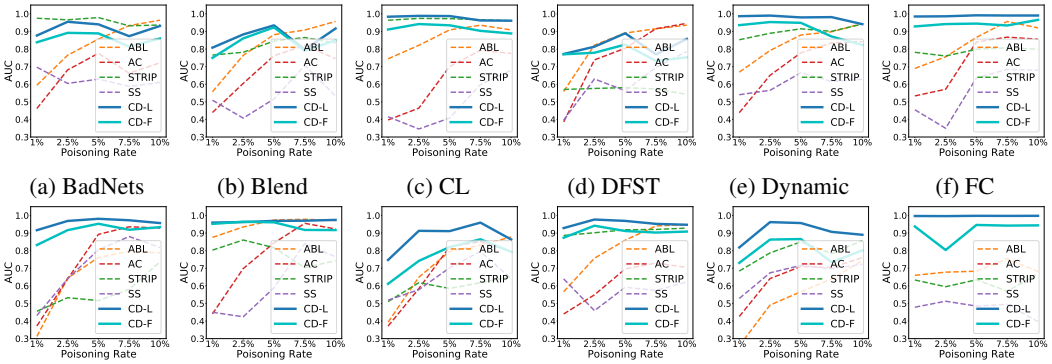
4.2 Robust Detection under Different Poisoning Rates

Here, we consider more challenging settings with varying poisoning rates [1%, 2.5%, 5%, 7.5%, 10%]. All other experimental settings are the same as in Section 4.1. Note that, under 1% poisoning rate, most of the attacks can still achieve an ASR $\geq 90\%$.

The results on the training sets are presented in Figure 3. Note that an AUC$= 0.5$ corresponds to random guessing, whereas an AUC< 0.5 suggests an opposite prediction. Similar results on the test sets can be found in Appendix C. It is evident that our method can achieve good detection performance robustly under different poisoning rates, except for a slight performance drop on Bend, DFST, Smooth, WaNet under high poisoning rates $> 5\%$. It can also be observed that ABL, AC, and SS are quite sensitive to the poisoning rate, but STRIP is comparably better. Only our methods and AC can successfully detect the more advanced attack ISSBA.

4.3 Improving Backdoor Mitigation

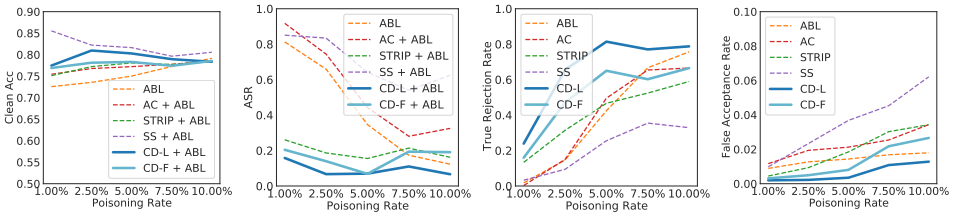
Next, we demonstrate that effective backdoor sample detection can help improve backdoor mitigation methods. We follow one of the state-of-the-art mitigation methods ABL (Li et al., 2021a)



(a) BadNets (b) Blend (c) CL (d) DFST (e) Dynamic (f) FC

(g) Nashville (h) SIG (i) Smooth (j) Trojan (k) WaNet (l) ISSBA

Figure 3: The detection performance (AUC) on CIFAR-10 training set under different poisoning rates (1%, 2.5%, 5%, 7.5%, 10%). Results are averaged across all 5 models following the same settings as in Table 1.



(a) CA (↑) (b) ASR (↓) (c) TRR (↑) (d) FAR (↓)

Figure 4: (a): Clean accuracy (CA) after mitigation; (b): Attack success rate (ASR) after mitigation; (c): True rejection rate (TRR); (d): False acceptance rate (FAR). All the experiments are averaged across all 5 models and 11 attacks on CIFAR-10 following the same setting as in Section 4.1.

to unlearn triggers from backdoored models using the detected backdoor subset $\mathcal{D}_b'$, and the clean subset $\mathcal{D}_c'$. ABL maximizes the model's loss on $\mathcal{D}_b'$ to remove the trigger while minimizing the loss on $\mathcal{D}_c'$ to maintain high clean accuracy. Based on the L_1 norm of the mask, here we select 2.5% samples with lower L_1 norm into $\mathcal{D}_b'$, and 70% samples with higher L_1 norm into $\mathcal{D}_c'$. We replace the backdoor isolation (detection) strategy in ABL with our two CD methods and denote the new methods as CD-L + ABL and CD-F + ABL, respectively. We run the experiments on all the attacks and models evaluated in Section 4.1. Detailed settings can be found in Appendix B.5.

The results are summarized in Figure 4. As can be observed, our CD-L method achieves the highest TRR and the lowest FAR, which both indicate good detection performance. Higher TRR means more backdoor samples are detected into $\mathcal{D}_b'$, facilitating more effective trigger unlearning. On the other hand, lower FAR means there are fewer backdoor samples left in $\mathcal{D}_c'$, which improves clean accuracy. Therefore, after mitigation, our CD-L method achieves the lowest ASR which is followed by our CD-F method. Note that the defender could also retrain the model on $\mathcal{D}_c'$, and in this case, FAR is the poisoning rate in $\mathcal{D}_c'$, i.e., lower is better. The above results indicate that accurate backdoor sample detection can provide substantial improvement to backdoor mitigation.

5 A Case Study on Bias Detection

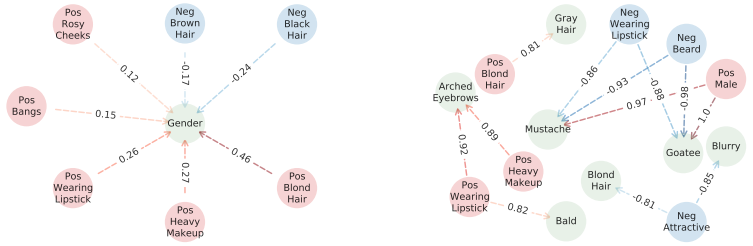
In this section, we show the connections between dataset bias and poisoning backdoor attacks. Recent studies have shown that curated datasets may contain potential biases (Torralba & Efros, 2011), which will be memorized by DNN models trained on such datasets. It has also been found that bias may be strongly associated with a particular class. For example, gender is associated with occupation (Bolukbasi et al., 2016) or facial attributes (Zhang et al., 2018; Tartaglione et al., 2021). On the other hand, DNNs tend to learn more of the "easier" correlations (potential biases) in the dataset and this usually happens at an earlier training stage, as captured by the sample-wise training loss (Nam et al., 2020; Du et al., 2021). Such a phenomenon was also observed and leveraged in ABL

(Li et al., 2021a) to isolate and mitigate backdoor attacks. These connections motivate us to apply our CD method to detect potential biases in a dataset. Here, we consider a real-world face dataset CelebA (Liu et al., 2015), which contains 162,770 face images and 40 binary facial attributes (the full training set is denoted as D). We train a classifier f_i for each facial attribute i, based on a shared feature extractor (ResNet-18). The detailed setup is described in Appendix B.9.

We apply CD-L to select 0.5% samples with the lowest $\|\boldsymbol{m}\|_1$ for each classifier f_i into a subset D'_i. We define the bias score as the distribution shift of attribute j from D to D'_i:

$$s(j,i) = (P_{ij} - P_j)/max(P_j, 1 - P_j), \tag{5}$$

where, $P_{ij} = |D'_{ij}|/|D'_i|$ is the percentage of samples in D'_i that have attribute j, $P_j = |D_j|/|D|$ is the percentage of samples in the full training set that have attribute j. $s(j,i)$ measures to what degree attribute j is predictive of attribute i (according to f_i). A positive/negative $s(j,i)$ value means that (positive/negative) attribute j is predictive of attribute i, closer to +1/-1 means more predictive. The $s(j,i)$ score allows us to identify the most predictive attributes to an attribute of interest, and potential biases if the absolute score is exceptionally high.



(a) Predictive attributes of *gender* attribute (b) All highly correlated attributes

Figure 5: Potential biases detected from the CelebA dataset using our CD-L method. Each node in the graph is an attribute, with the links pointing (with score $s(j,i)$) from a highly predictive attribute to the attribute of interest (green nodes). Red/blue nodes and links represent positive/negative predictive attributes. Darker color edge indicates larger absolute $s(j,i)$ score.

As shown in Figure 5a, applying CD-L on the gender classifier, we identify the top-2 most predictive attributes of *gender*: *blond hair* and *heavy makeup*. This matches with one well-known bias in CelebA, i.e., *gender* is mostly determined by (Nam et al., 2020; Tartaglione et al., 2021). Furthermore, our method also reveals two potential negative biases: *brown hair* and *black hair*, meaning that gender is hard to predict when these two hair colors appear. It also indicates that blond hair is mutually exclusive with other hair colors. Figure 5b further shows all the highly correlated (either positive or negative) attributes identify from CelebA, where the absolute $s(j,i)$ score are all greater than 0.8. As suggested by Zhang et al. (2018), defining ground-truth biases is subject to different human interpretations. Several connections revealed in Figure 5b are closely related to known human stereotypes, such as male with different kinds of beard, or makeup with arched eyebrows. The above results demonstrate the possibility of our method to help detect potential biases in real-world datasets, a challenging problem needs to be tackled for fair machine learning.

6 Conclusion

In this paper, we proposed a novel method, *Cognitive Distillation* (CD), that extracts the minimal pattern responsible for the model's prediction. With CD and the distilled patterns, we reveal that backdoored models make backdoor predictions only based on a sparse pattern of the input, regardless the large trigger patterns used by different attacks. And the location of the pattern reveals the core part of the trigger. This allow us to build simple but effective backdoor sample detectors based on the learned input masks. We empirically show, with 6 DNN architectures, 3 datasets and 12 advanced backdoor attacks that our proposed detection method is both effective and robust (under different poising rates). Our CD methods can also help detect potential biases in real-world datasets. The unsupervised nature of CD makes it a generic technique for investigating the underlying inference mechanism of different types of DNNs.

REFERENCES

Eugene Bagdasaryan and Vitaly Shmatikov. Blind backdoors in deep learning models. In *USENIX Security*, 2021.

Mauro Barni, Kassem Kallas, and Benedetta Tondi. A new backdoor attack in cnns by training set corruption without label poisoning. In *ICIP*, 2019.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *NeurIPS*, 2016.

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.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *NeurIPS*, 2020.

Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *WACV*, 2018.

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

Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks. In *IJCAI*, 2019.

Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

Siyuan Cheng, Yingqi Liu, Shiqing Ma, and Xiangyu Zhang. Deep feature space trojan attack of neural networks by controlled detoxification. *arXiv preprint arXiv:2012.11212*, 2020.

Edward Chou, Florian Tramer, and Giancarlo Pellegrino. Sentinet: Detecting localized universal attacks against deep learning systems. In *SPW*, 2020.

Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. *NeurIPS*, 2017.

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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

Khoa Doan, Yingjie Lao, and Ping Li. Backdoor attack with imperceptible input and latent modification. *NeurIPS*, 2021a.

Khoa Doan, Yingjie Lao, Weijie Zhao, and Ping Li. Lira: Learnable, imperceptible and robust backdoor attacks. In *ICCV*, 2021b.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

Mengnan Du, Varun Manjunatha, Rajiv Jain, Ruchi Deshpande, Franck Dernoncourt, Jiuxiang Gu, Tong Sun, and Xia Hu. Towards interpreting and mitigating shortcut learning behavior of nlu models. In *NAACL-HLT*, 2021.

Yu Feng, Benteng Ma, Jing Zhang, Shanshan Zhao, Yong Xia, and Dacheng Tao. Fiba: Frequency-injection based backdoor attack in medical image analysis. In *CVPR*, 2022.

Ruth Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *ICCV*, 2017.

Ruth Fong, Mandela Patrick, and Andrea Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *ICCV*, 2019.

Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. Strip: A defence against trojan attacks on deep neural networks. In *ACSAC*, 2019.

Siddhant Garg, Adarsh Kumar, Vibhor Goel, and Yingyu Liang. Can adversarial weight perturbations inject neural backdoors. In *CIKM*, 2020.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2020.

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

Jiyang Guan, Zhuozhuo Tu, Ran He, and Dacheng Tao. Few-shot backdoor defense using shapley estimation. In *CVPR*, 2022.

Junfeng Guo, Ang Li, and Cong Liu. Aeva: Black-box backdoor detection using adversarial extreme value analysis. *arXiv preprint arXiv:2110.14880*, 2021.

Wenbo Guo, Lun Wang, Xinyu Xing, Min Du, and Dawn Song. Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems. *arXiv preprint arXiv:1908.01763*, 2019.

Jonathan Hayase, Weihao Kong, Raghav Somani, and Sewoong Oh. Defense against backdoor attacks via robust covariance estimation. In *ICML*, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, 2016b.

Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. Detection of traffic signs in real-world images: The german traffic sign detection benchmark. In *IJCNN*, 2013.

Xiaoling Hu, Xiao Lin, Michael Cogswell, Yi Yao, Susmit Jha, and Chao Chen. Trigger hunting with a topological prior for trojan detection. In *ICLR*, 2022.

Kunzhe Huang, Yiming Li, Baoyuan Wu, Zhan Qin, and Kui Ren. Backdoor defense via decoupling the training process. In *ICLR*, 2022.

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *NeurIPS*, 32, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash, and Heiko Hoffmann. Universal litmus patterns: Revealing backdoor attacks in cnns. In *CVPR*, 2020.

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

Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Anti-backdoor learning: Training clean models on poisoned data. *NeurIPS*, 2021a.

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*, 2021b.

Yiming Li, Tongqing Zhai, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shutao Xia. Rethinking the trigger of backdoor attack. *arXiv preprint arXiv:2004.04692*, 2020.

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

Junyu Lin, Lei Xu, Yingqi Liu, and Xiangyu Zhang. Composite backdoor attack for deep neural network by mixing existing benign features. In *CCS*, 2020.

Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *RAID*, 2018a.

Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. 2018b.

Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. Abs: Scanning neural networks for back-doors by artificial brain stimulation. In *CCS*, 2019.

Yingqi Liu, Guangyu Shen, Guanhong Tao, Zhenting Wang, Shiqing Ma, and Xiangyu Zhang. Complex backdoor detection by symmetric feature differencing. In *CVPR*, 2022.

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

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

Aravindh Mahendran and Andrea Vedaldi. Salient deconvolutional networks. In *ECCV*, 2016.

Naren Sarayu Manoj and Avrim Blum. Excess capacity and backdoor poisoning. In *NeurIPS*, 2021.

Junhyun Nam, Hyuntak Cha, Sungsoo Ahn, Jaeho Lee, and Jinwoo Shin. Learning from failure: De-biasing classifier from biased classifier. *NeurIPS*, 2020.

Tuan Anh Nguyen and Anh Tran. Input-aware dynamic backdoor attack. *NeurIPS*, 2020.

Tuan Anh Nguyen and Anh Tuan Tran. Wanet - imperceptible warping-based backdoor attack. In *ICLR*, 2021.

Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. *arXiv preprint arXiv:1806.07421*, 2018.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?" explaining the predictions of any classifier. In *SIGKDD*, 2016.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017.

Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. *NeurIPS*, 2018.

Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *CCS*, 2016.

Guangyu Shen, Yingqi Liu, Guanhong Tao, Shengwei An, Qiuling Xu, Siyuan Cheng, Shiqing Ma, and Xiangyu Zhang. Backdoor scanning for deep neural networks through k-arm optimization. In *ICML*, 2021.

Ilia Shumailov, Zakhar Shumaylov, Dmitry Kazhdan, Yiren Zhao, Nicolas Papernot, Murat A Erdogdu, and Ross Anderson. Manipulating sgd with data ordering attacks. *NeurIPS*, 2021.

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

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019.

Di Tang, XiaoFeng Wang, Haixu Tang, and Kehuan Zhang. Demon in the variant: Statistical analysis of {DNNs} for robust backdoor contamination detection. In *USENIX Security*, 2021.

Enzo Tartaglione, Carlo Alberto Barbano, and Marco Grangetto. End: Entangling and disentangling deep representations for bias correction. In *CVPR*, 2021.

Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *CVPR*, 2011.

Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. *NeurIPS*, 2018.

Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Clean-label backdoor attacks. 2018.

Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *S&P*, 2019.

Ren Wang, Gaoyuan Zhang, Sijia Liu, Pin-Yu Chen, Jinjun Xiong, and Meng Wang. Practical detection of trojan neural networks: Data-limited and data-free cases. In *ECCV*, 2020.

Emily Wenger, Josephine Passananti, Arjun Nitin Bhagoji, Yuanshun Yao, Haitao Zheng, and Ben Y Zhao. Backdoor attacks against deep learning systems in the physical world. In *CVPR*, 2021.

Dongxian Wu and Yisen Wang. Adversarial neuron pruning purifies backdoored deep models. In *NeurIPS*, 2021.

Zhen Xiang, David Miller, and George Kesidis. Post-training detection of backdoor attacks for two-class and multi-attack scenarios. In *ICLR*, 2022.

Xiaojun Xu, Qi Wang, Huichen Li, Nikita Borisov, Carl A Gunter, and Bo Li. Detecting ai trojans using meta neural analysis. In *S&P*, 2021.

Kaiwen Yang, Tianyi Zhou, Xinmei Tian, Dacheng Tao, et al. Class-disentanglement and applications in adversarial detection and defense. *NeurIPS*, 2021.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

Yi Zeng, Won Park, Z Morley Mao, and Ruoxi Jia. Rethinking the backdoor attacks' triggers: A frequency perspective. In *ICCV*, 2021.

Yi Zeng, Si Chen, Won Park, Zhuoqing Mao, Ming Jin, and Ruoxi Jia. Adversarial unlearning of backdoors via implicit hypergradient. In *ICLR*, 2022.

Quanshi Zhang, Wenguan Wang, and Song-Chun Zhu. Examining cnn representations with respect to dataset bias. In *AAAI*, 2018.

Pu Zhao, Pin-Yu Chen, Payel Das, Karthikeyan Natesan Ramamurthy, and Xue Lin. Bridging mode connectivity in loss landscapes and adversarial robustness. In *ICLR*, 2020a.

Shihao Zhao, Xingjun Ma, Xiang Zheng, James Bailey, Jingjing Chen, and Yu-Gang Jiang. Clean-label backdoor attacks on video recognition models. In *CVPR*, pp. 14443–14452, 2020b.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016.

A Difference of CD to Existing Methods

Difference with Model Interpretation Methods. Our CD can provide a certain degree of explanation about how the model makes its predictions, thus is related to model interpretation methods. Gradient-based interpretation methods generate saliency maps (Simonyan et al., 2013), such as deconvolution (Zeiler & Fergus, 2014) and guided backpropagation (Springenberg et al., 2014; Mahendran & Vedaldi, 2016) to help interpret how the model makes predictions given an input image. Perturbation-based methods (Dabkowski & Gal, 2017; Petsiuk et al., 2018; Fong et al., 2019; Yang et al., 2021) are a generalized version of saliency maps that integrate several rounds of backpropagation to learn an explanation (Fong & Vedaldi, 2017). Meaningful Perturbations (MP) (Fong & Vedaldi, 2017) generate interpretations via deletion. It searches for a perturbed input that maximizes the score for a specific class. LIME (Ribeiro et al., 2016) is one of the foundational works for interpretation. It trains an additional locally faithful interpretable model to explain the target model. E.g., the faithful model can be a linear model that locally behaves the same as the target model. Similar to LIME, CD also emphasizes faithfulness, but it does not involve an additional model. Another set of methods is Class Activation Mapping (CAM) (Zhou et al., 2016) and its variants (Selvaraju et al., 2017; Chattopadhay et al., 2018). CAM-based methods use the gradient of the target concept (label) through a specific layer and generate a coarse localization map highlighting important regions.

Technical Comparison. The difference of our CD from closely-related methods, including LIME (Ribeiro et al., 2016), Meaningful Perturbation (MP) (Fong & Vedaldi, 2017), and Neural Cleanse (NC) (Wang et al., 2019), are summarized in Table 2. Different from LIME, MP and NC, our CD method is unsupervised, that is, the distillation process does not rely on the class labels. And CD searches for the most determining input pattern of the model output, for whichever class it predicts.

Table 2: The objective functions of LIME, Meaningful Perturbation (MP), Neural Cleanse (NC) and our Cognitive Distillation (CD). $\mathcal{L}_{CE}$ is the cross-entropy loss.

Method	Optimization Objective
LIME	$\arg\min_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g)$
NC	$\arg\min_{\boldsymbol{m}, \boldsymbol{\Delta}, t \in 1 \dots K} \mathcal{L}_{CE}(f((1-\boldsymbol{m}_t) \odot \boldsymbol{x} + \boldsymbol{m}_t \odot \boldsymbol{\Delta}_t), y_t) + \alpha \lVert \boldsymbol{m}_t \rVert_1$
MP	$\arg\min_{\boldsymbol{m}} \mathcal{L}_{CE}(f(\boldsymbol{x} \odot \boldsymbol{m} + (1-\boldsymbol{m}) \odot \phi), y_t)) + \alpha \lVert 1 - \boldsymbol{m} \rVert_1$
CD (ours)	$\arg\min_{\boldsymbol{m}} \lVert f(\boldsymbol{x}) - f(\boldsymbol{x} \odot \boldsymbol{m} + (1-\boldsymbol{m}) \odot \delta) \rVert_1 + \alpha \lVert \boldsymbol{m} \rVert_1 + \beta TV(\boldsymbol{m})$

Detailed Difference with LIME. LIME aims to find an additional interpretable model g, with $\Omega(g)$ regularizing its complexity. $\mathcal{L}$ enforces the local fidelity that g behaves faithfully to f (DNN) around $\boldsymbol{x}$ (using the mask to perturb around $\boldsymbol{x}$). When applying LIME on images, ridge regression is selected as g, and one requires a targeted class y_t intended to explain. The output is a coarse heatmap defined by fixed super-pixels. However, our method is different from LIME, and there is no additional model g or targeted class y_t involved. Also, our generated masks are fine-grained with the most important pixels for the model's prediction.

Detailed Difference with NC. NC optimizes for both a mask and the trigger pattern. The optimization outputs a mask $\boldsymbol{m}_t$ and a trigger pattern Δ_t for a class y_t. The objective of NC is to disentangle a perturbed input image $\boldsymbol{x}'$ into a class-wise universal trigger pattern Δ_t with a corresponding mask $\boldsymbol{m}_t$ and residual image $(1-\boldsymbol{m}_t) \odot \boldsymbol{x}$. For $\boldsymbol{x} \in \mathcal{D}$, the perturbed input $\boldsymbol{x}'$ can make the model predict the target class y_t. NC can be used for backdoored label detection, trigger synthesis (recovery) and backdoor model detection. It measures the L_1 norm of the mask $\boldsymbol{m}_t$ for all classes and the Median Absolute Deviation for detecting backdoored models and labels, rather than backdoor samples. Also note that existing works (Liu et al., 2019; Nguyen & Tran, 2020; Li et al., 2021c; Cheng et al., 2020) have demonstrated that advanced attacks can still evade NC's detection. As discussed in NC (Wang et al., 2019), it is also not efficient for models/datasets with a large number of classes/labels.

Different with NC, our CD aims to find the pattern for each sample in the dataset rather a class of samples. Both CD's output and main optimization functions are different from NC: 1) CD outputs sample-specific masks and patterns, 2) its objective is the *unsupervised* absolute difference between $f(\boldsymbol{x})$ and $f(\boldsymbol{x}')$, rather than the *supervised* cross-entropy loss ($\mathcal{L}_{CE}$). Moreover, CD's computational complexity does not increase with the number of classes.

Detailed Difference with MP. MP aims to find a perturbation mask that can explain a target class y_t through $\mathcal{L}_{CE}$. The target class is also known as a *concept* that needs to be explained by the model. The deleted perturbation operator ϕ includes constant, noise, and blur. MP aims to find deletion regions that are maximally informative (for y_t), while CD aims to find regions that can be preserved for the prediction (and does not require y_t). For the removed pixel, CD uses a random color value δ in each optimization step, while the ϕ is the same operation applied in every step.

MP is more similar to NC than our CD, since both of them use CE as the main objective. The backdoor class label of the image must be known in advance for backdoor sample detection.

In a nutshell, MP asks the model *what can be removed to make a cat image not like a 'cat'?*, however, our CD asks *what is the minimal sufficient pattern the model needs to make its prediction?*

B Experimental settings

All experiments are run with NVIDIA Tesla P100/V100/A100 GPUs with PyTorch implementations. Sample code is available in the supplementary materials and will be open-sourced. For CIFAR10 experiments, we train for 60 epochs with SGD optimizer, weight decay 5×10^{-4}, initial learning rate 0.1 decay by 0.1 at the 45th epoch. For GTSRB, we use the same set up with CIFAR10, except for 40 epochs and decay at the 35th epcoh. For the ImageNet subset, we train 200 epochs with the same learning rate settings and weight decays, except the learning rate adopts a cosine decay without restart (Loshchilov & Hutter, 2017).

B.1 Evaluations

For test-time detection, we report the True Positive Rate (TPR) and False Positive Rate (FPR) using one standard deviation ($\gamma = 1$) lower than the mean calculated on 1% of the training subset as the threshold t. Results can be found in Appendix C. We also use test-set clean accuracy (CA) and attack success rate (ASR) to verify the effectiveness of the attacks.

B.2 Datasets

The datasets used in our experiments in Section 4.1 are summarized in Table 3.

Table 3: Statistics the datasets used in our experiments

Dataset	No. Classes	Input Size	No. Training Images
CIFAR10 (Krizhevsky et al., 2009)	10	32×32×3	50,000
GTSRB (Houben et al., 2013)	43	32×32×3	39,209
ImageNet subset (Deng et al., 2009)	200	224×224×3	100,000

B.3 Attack Configurations

Table 4: Configurations and implementation details of the backdoor attacks.

Attacks	Trigger Size	Trigger Pattern	Trigger Type	Strategy
BadNets (Gu et al., 2017)	Patch	Grid	Class-wise	Pixel modification
Blend (Chen et al., 2017)	Full image	Hello Kitty	Class-wise	Blend in
CL (Turner et al., 2018)	Full image	Patch + Adv Noise	Class-wise	Optimizations
DFST (Cheng et al., 2020)	Full image	Style transfer	Sample-specific	GAN
Dynamic (Nguyen & Tran, 2020)	Few pixels	Mask generator	Sample-specific	Mask Generator
FC (Shafahi et al., 2018)	Full image	Optimized Noise	Sample-specific	Optimization
SIG (Barni et al., 2019)	Full image	Periodical signal	Class-wise	Blend in
Trojan (Liu et al., 2018b)	Patch	Watermark	Class-wise	Pixel modification
WaNet (Nguyen & Tran, 2021)	Full image	Invisible noise	Sample-specific	Wrapping function
Nashville (Liu et al., 2019)	Full image	Style transfer	Sample-specific	Instagram filter
Smooth (Zeng et al., 2021)	Full image	Style transfer	Sample-specific	Smooth frequency space
ISSBA (Li et al., 2021c)	Full image	Mask Generator	Sample-specific	Encoder

The detailed configurations of the 12 backdoor attacks considered in our experiments are summarized in Table 4. We have verified their attack success rate (ASR) and clean accuracy (CA) on the

test sets, and all the attacks are effective as shown in Section 4.1. Visualizations of the backdoor samples are in Appendix D. For WaNet (Nguyen & Tran, 2021), we find its original hyperparameters are not strong enough to generate effective triggers for data-poisoning. We thus increase the uniform grid size to 32×32 to improve its ASR. Class-wise pattern means to apply the same pattern on all images of a particular class, while sample-specific pattern means that each sample has a specific pattern.

B.4 Detection Configurations

Table 5: Detection configurations of backdoor sample detection methods. For all baseline methods, we followed their official open-sourced implementations.

Method	Detection confidence score
ABL (Li et al., 2021a)	Training loss of first 20 epochs
AC (Chen et al., 2018)	Silhouette scores
STRIP (Gao et al., 2019)	Entropy
SS (Tran et al., 2018)	Top right singular vector

Configurations of the sample detection methods used in the experiments are summarized in Table 5. ABL is a hybrid method of both backdoor sample detection and backdoor mitigation. Here we only use its sample detection part, which was referred to as backdoor sample isolation in the paper (Li et al., 2021a). We use the sample-specific training loss averaged over the first 20 epochs of model training as the confidence score. For AC, SS and STRIP, they are specifically designed for poison sample detection, and we use their original settings. For SS, we use the top right singular vector of the SVD decomposition as the confidence score. In our threat model, it assumes that the defender does not have prior knowledge of which samples are clean, which is a more realistic in real-world scenarios. For STRIP, we use all training data as the sampling pool for superimposing as the defender does not have any clean samples. Interestingly, we find this does not dramatically affect SRTIP's performance.

For test-time detection, except for STRIP, other baseline methods cannot be directly applied. This is because both AC and SS require ground truth label y, for which we replace it with the model's prediction. It should have minimal impact on the detection performance if the prediction is accurate, should degrade the performance otherwise. We exclude ABL from our test-time detection experiments, as it requires training statistics, and there is no intuitive solution to accommodate this requirement. We assume all the methods can calculate the mean score on 1% of the clean training set and use 1 standard deviation lower than the mean as the threshold.

CD. The β is set to 10. The α is set to 0.01/0.001 for models trained on CIFAR10 and GTSRB, 0.001/0.0001 for the ImageNet subset model, for CD-L/CD-F, respectively. We use Adam optimizer (Kingma & Ba, 2014) with initial learning rate 0.1, β_1=0.1, β_2=0.1, and a total of 100 steps to learn the input mask. We use the scaled tanh function to ensure the mask is between [0, 1].

B.5 Backdoor Mitigation

Following previous work (Li et al., 2021a), the defender can take a step further to remove the backdoor via a process of anti-backdoor finetuning. I.e., finetuning the backdoored model to maximizing its loss on $\mathcal{D}_b'$ while minimizing its loss on $\mathcal{D}_c'$, as following:

$$\arg\min_{\theta} \mathbb{E}_{(\boldsymbol{x}_c, y_c)\sim\mathcal{D}_c'} \mathcal{L}(f_\theta(\boldsymbol{x}_c), y_c) + \arg\max_{\theta} \mathbb{E}_{(\boldsymbol{x}_b, y_b)\sim\mathcal{D}_b'} \mathcal{L}(f_\theta(\boldsymbol{x}_b), y_b). \tag{6}$$

We use the method described in Algorithm 1. We set p_b to 2.5% and p_c to 70%, and optimize for 10 epochs with learning rate set to 5×10^{-4}.

B.6 Ablation Study of the Regularization Terms

The goal of our CD is to find the most determining pattern from the input image of the model's prediction via a learnable mask. Intuitively, one can use following objective function:

$$\arg\min_{\boldsymbol{m}} \|f_\theta(\boldsymbol{x}) - f_\theta(\boldsymbol{x} \odot \boldsymbol{m})\|_1 + \alpha\|(\boldsymbol{m})\|_1, \tag{7}$$

Algorithm 1 Unlearning and Fine-tuning

Input: f_θ, $\mathcal{D}$, percentage p_b for $\mathcal{D}'_b$, percentage p_c for $\mathcal{D}'_c$, Scores $\|\boldsymbol{m}\|_1$, Epochs E
$\mathcal{D}$ = argsort($\mathcal{D}$, s)
$\mathcal{D}'_b$ = select p_b samples with lower s from $\mathcal{D}$ ▷ detected as backdoor samples
$\mathcal{D}'_c$ =select p_c samples with higher s from $\mathcal{D}$ ▷ detected as clean samples
for e **to** E **do**
  optimize $\arg\min_\theta \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}'_c}\mathcal{L}(f_\theta(\boldsymbol{x}), y)$
  optimize $\arg\max_\theta \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}'_b}\mathcal{L}(f_\theta(\boldsymbol{x}), y)$
end for
Output: f_θ

where, $\boldsymbol{x} \odot \boldsymbol{m}$ is the perturbed input and the unimportant pixels are replaced with 0. The above objective may not be ideal as there may exist black/darker pixels with values very close to 0, which will be surely removed by the $\|\boldsymbol{m}\|_1$ regularization. However, in clean images, black/darker pixels may also be important. As shown in Figure 6, the optimized mask only considers the white pixels of the trigger are necessary for the model's prediction, with the darker pixels are completely ignored. For the clean samples, darker pixels of the main objects ("horse" or "peacock") are also important.

In order to address the above issue, we replace the unimportant pixels with a random noise vector δ sampled at each optimization step, using the following objective function:

$$\arg\min_{\boldsymbol{m}} \|f_\theta(\boldsymbol{x}) - f_\theta(\boldsymbol{x} \odot \boldsymbol{m} + (1-\boldsymbol{m}) \odot \delta)\|_1 + \alpha\|(\boldsymbol{m})\|_1, \tag{8}$$

where, random noise δ is applied to all spatial dimensions. This objective function will force the mask to consider darker pixels as equally important for the model's prediction as white pixels. As shown in Figure 7, the optimized mask now can reveal the black pixels of the backdoor trigger. For clean samples, darker pixels of the main objects ("horse" and "peacock") can now also be considered as important by our CD.

In order to make the extracted mask to be smooth and more interpretable, we add one more regularization term to CD's objective:

$$\arg\min_{\boldsymbol{m}} \|f_\theta(\boldsymbol{x}) - f_\theta(\boldsymbol{x} \odot \boldsymbol{m} + (1-\boldsymbol{m}) \odot \delta)\|_1 + \alpha\|(\boldsymbol{m})\|_1 + \beta TV(\boldsymbol{m}), \tag{9}$$

where, the TV is the total variation loss and β balances the TV with other objectives. This ensures the model to consider the surrounding pixels of an important pixel as similarly important. As shown in Figure 8, the learned mask with the TV regularization can outline the shape of the main object.

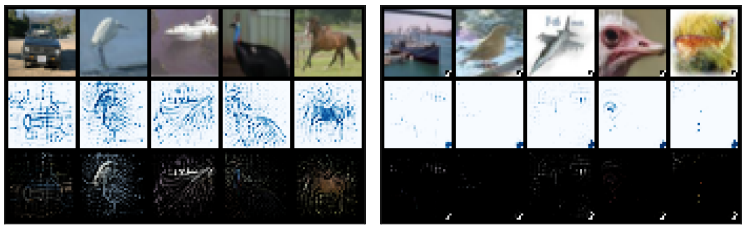
(a) Clean samples (b) Backdoor samples

Figure 6: First row: examples of clean and backdoor images (BadNets) randomly selected from the CIFAR10 training set. Second row: the corresponding learned masks $\boldsymbol{m}$ by using Equation 7. Third row: distilled pattern $\boldsymbol{x} \odot \boldsymbol{m}$ for the corresponding masks and images.

B.7 Ablation Study of the Hyperparameters

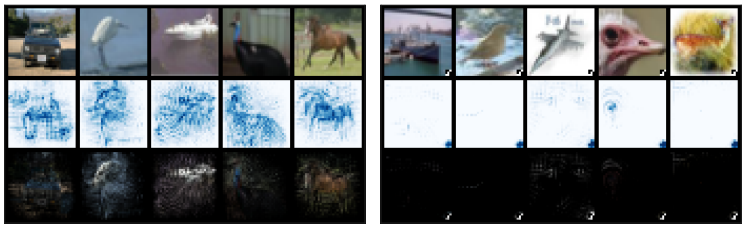
In this section, we perform an ablation study for the sensitivity of each hyperparameter for backdoor detection performance.

Sensitivity of α on detection performance. The α controls the number of pixels that can be kept in the distilled pattern. If too many or too few pixels are removed from the pattern, then it would



(a) Clean samples (b) Backdoor samples

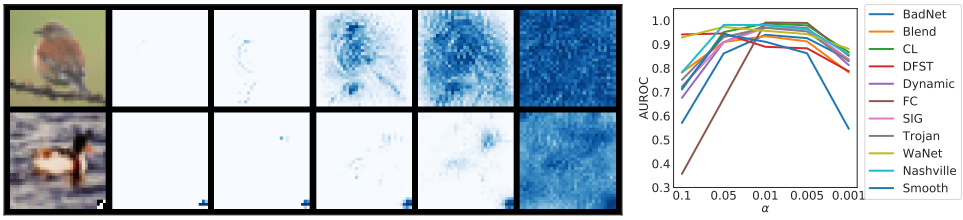
Figure 7: Learned masks $\boldsymbol{m}$ is using Equation 8.



(a) Clean samples (b) Backdoor samples

Figure 8: Learned masks $\boldsymbol{m}$ is using Equation 9.

be difficult to distinguish between the backdoor and clean samples. α should be set in a way that the extracted pattern mainly focuses on the main object in the image. In practice, the defender may generate patterns for a small subset of samples to examine the effect of the pattern. As shown in Figure 9a, $\alpha = 0.01$ shows the extracted pattern focuses on the main object of the image, and the mask removes other pixels. From Figure 9b, it can also be observed that the detection performance is stable from 0.05 to 0.005 for all attacks.



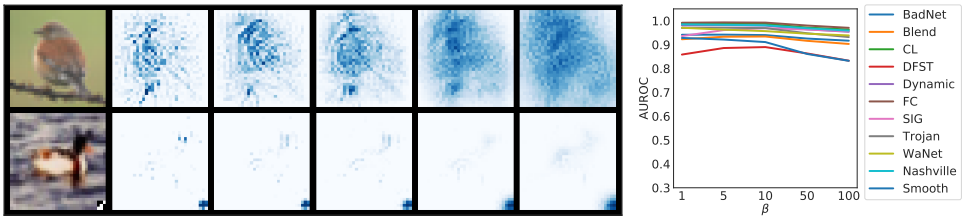
(a) Visualizations (b) AUC

Figure 9: Varying α in [0.1, 0.05, 0.01, 0005, 0.001] for CD. (a) Visualizations of the learned mask for clean (top row) and backdoor (bottom row) samples. (b) The detection performance of AUC on the training set for each attack with 5% poisoning rate. Results are averaged across models.

Sensitivity of β on detection performance. It can be observed that β controls how smooth is the learned mask. Assigning higher values to β will produce smoothed mask, while lower values β generate a sparser mask. The location and pattern of the essential pixels remain approximately the same. As shown in Figure 10a, a smoothed mask could produce a mask that matches the human understanding. As shown in Figure 10b, β will not significantly affect the backdoor detection performance.

B.8 Whitebox Adaptive Attacks

In this section, we evaluate the performance of our model in the whitebox scenario (i.e., where an attacker is aware of our detection model) and deploy adaptive strategies. In order to evade detection, adversaries need to sacrifice one of the two crucial components of backdoor attacks, including attack success rate and stealthiness.

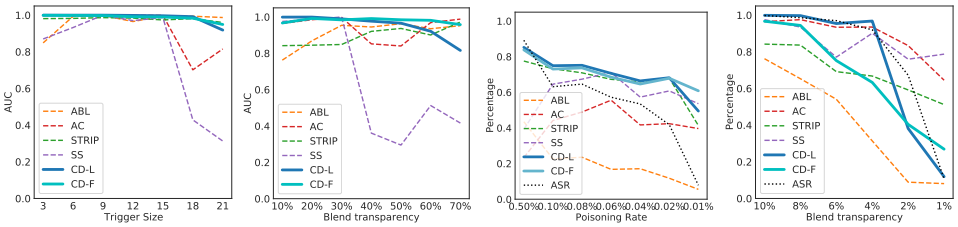


(a) Visualizations (b) AUC

Figure 10: Varying β in [1, 5, 10, 50, 100] for CD. (a) Visualizations of the learned mask for clean (top row) and backdoor (bottom row) samples. (b) The detection performance of AUC on the training set for each attack with 5% poisoning rate. Results are averaged across models.

Increase trigger visibility. CD uses the trigger's size perceived by the model as its detection metric. Several existing attacks evaluated in Table 6 already apply the trigger on the entire image, see a summary of these attacks in Appendix B.3. As shown in Figure 1, the model does not use the entire backdoored image for its predictions, even if the trigger is applied to the whole image. For adaptive strategy, the adversary should increase the size of the trigger perceived by the model, including 1) increase the size of the patched trigger, and 2) increase the visibility of the blended trigger (by reducing the transparency). We use BadNets and Blend attacks to demonstrate these strategies since they can be easily controlled by a hyperparameter.

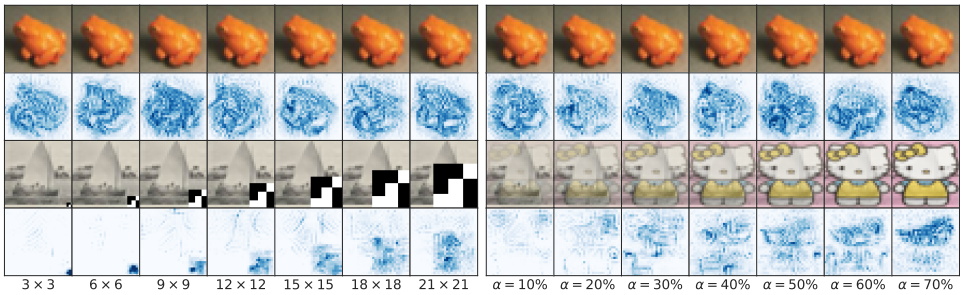
Decrease Attack Success Rate. CD also distilles pattern based on the model's response. The attacker may deliberately sacrifice ASR by using a weaker trigger that does not guarantee the attack's success. In other words, in some cases, the trigger is not responsible for the prediction. This can be achieved by 1) decreasing the poisoning rates, and 2) reducing the visibility of the trigger. We use BadNets, SIG, and Nashville attacks to reduce the poisoning rates to an extremely low level. We also evaluate the performance of the Blend attack by reducing the transparency.



(a) Increase trigger visibility. (b) Decrease attack success rate.

Figure 11: (a): Increasing trigger size for BadNets attack and the transparency for the Blend attack. (b): Decreasing poisoning rates to lower ASR (results averaged across BadNest, SIG and Nashville attacks), and the transparency of the Blend attack. All experiments evaluated on CIFAR10 with ResNet-18. Results for detection methods reported as AUC on the training set.

As shown in Figure 11a, if the size or the visibility of the trigger increases, it could reduce the detection performance of CD. However, this will greatly impact the stealthiness of the backdoor attacks. Visualization of the backdoor samples are shown in Figure 12. The trigger size needs to be as large as the main content in the image to evade detection. For decrease ASR shown in Figure 11b, the attacker could sacrifices some ASR to evade detection. For CD, it is comparably more robust towards decreasing poisoning rates to reduce ASR, but slightly less effective with blend transparency is 2%. Both CD and STRIP are based on input-output responses, and if the attacks are barely successful, then the detection performance could be affected. Nevertheless, in either case, the effectiveness (stealthiness or ASR) of the backdoor will be impacted.



(a) Increase the trigger size of the BadNets (b) Increase the trigger transparency of the Blend

Figure 12: First row: examples of clean images from the CIFAR10 training set. Second row: the corresponding learned masks of clean images extracted for each model trained under different trigger size and visibility. Third row: examples of the backdoor image with different trigger size and visibility. Forth row: the corresponding learned masks of backdoor images extracted for each model trained under different trigger size and visibility.

B.9 Bias Detection

Experiment setting. For all the 40 binary facial attributes, we use a ResNet-18 (He et al., 2016a) as the feature extractor and attach a classifier head for each attribute. All classifier uses cross-entropy loss and shares the feature extractor. We train the model with 10 epochs, with SGD optimizer, and weight decay is set to 5×10^{-4}. For CD, we set α to 0.001, β to 1000 and optimize the mask for 300 steps with learning rate 0.075.

Figure 13 shows a visualization of the learned mask using CD-L. It can be observed that the mask can locate the area for each facial attribute that matches a human's understanding, suggesting that CD-L provides a degree of explanation of the model's prediction.

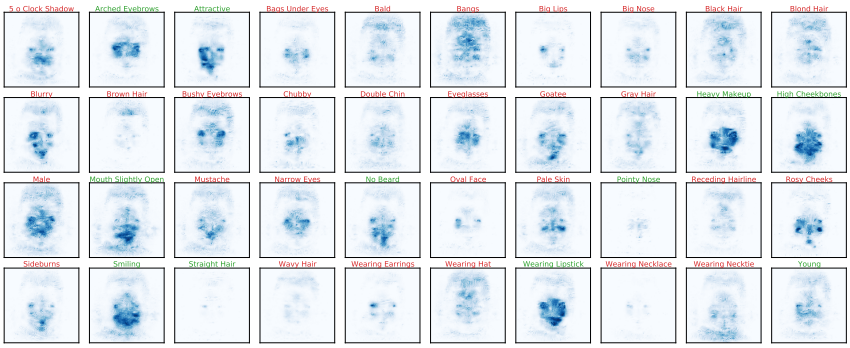


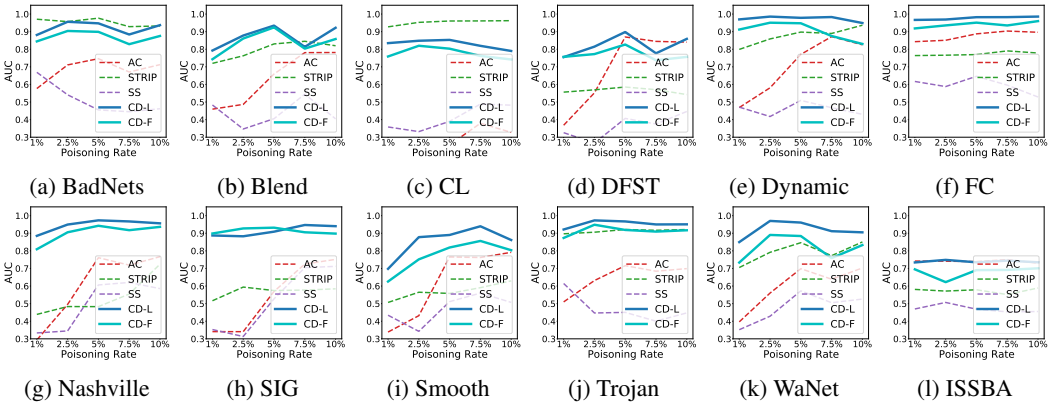
Figure 13: Visualization of optimized mask using CD-L for bias detection experiments. Positive facial attributes are indicated as green and red as negative.

C Additional Experiment Results

In this section, we show additional results, including the detection performance under varying poisoning rates on the test set in Figure 14, and detailed results that are summarized in Table 1 in the main paper. Detailed results for training and test set detection using AUC are in Table 6 and Table 7, respectively. The result of using TRP/FPR with one standard deviation ($\gamma = 1$) lower than the mean calculated on 1% of the training subset as the threshold is in Table 8.

Table 6: The detection performance on training set measured by AUC ((%)) for our CD method and the baselines against 12 backdoor attacks. The poisoning rate is 5% for all attacks. The Clean Accuracy (CA) is evaluated on clean test set. The best detection results are in **bold**.

Dataset	Attack	Model	CA	ASR	ABL	AC	STRIP	SS	CD-L	CD-F
CIFAR10	BadNets	VGG-16	89.05	100.00	88.31	64.41	94.82	21.41	**99.98**	99.96
		RN-18	87.60	100.00	85.87	99.97	98.68	99.48	**100.00**	99.99
		MobileV2	84.19	100.00	72.70	91.38	98.15	67.42	**99.73**	94.88
		PARN-101	89.77	100.00	83.58	45.05	98.65	**99.58**	74.10	67.69
		GoogLeNet	92.84	100.00	97.74	87.05	**99.13**	26.54	96.34	81.92
	Blend	VGG-16	89.26	99.60	90.50	65.05	76.37	22.58	97.46	**97.76**
		RN-18	87.77	100.00	88.78	98.80	78.67	96.20	**99.92**	96.20
		MobileV2	84.81	99.40	85.69	46.56	88.08	59.43	**98.40**	84.91
		PARN-101	89.66	99.8 0	91.54	75.69	90.76	41.33	**97.89**	94.03
		GoogLeNet	92.39	99.20	84.32	**95.04**	88.86	38.61	73.66	88.62
	CL	VGG-16	89.00	98.40	89.88	80.19	92.76	46.52	**99.99**	99.80
		RN-18	87.33	98.40	89.95	81.91	97.91	39.15	**99.97**	98.44
		MobileV2	85.15	90.00	82.95	59.28	98.02	34.46	**98.90**	89.82
		PARN-101	89.65	93.60	94.75	38.35	**98.90**	44.45	95.60	88.08
		GoogLeNet	92.69	53.60	96.77	90.57	98.75	39.31	**99.25**	91.26
	DFST	VGG-16	88.09	100	81.83	37.90	61.93	6.73	92.53	**95.59**
		RN-18	87.34	100.00	96.31	97.12	57.89	**98.73**	98.42	95.52
		MobileV2	84.67	99.80	80.41	**84.47**	61.07	38.40	81.15	68.65
		PARN101	89.37	99.60	**94.91**	83.83	57.56	42.42	76.85	85.46
		GoogLeNet	92.90	100.00	92.04	**98.93**	51.95	95.39	95.83	67.49
	Dynamic	VGG-16	88.94	100.00	93.12	54.18	80.94	48.16	**99.86**	99.14
		RN-18	87.93	100.00	96.83	95.55	84.98	93.44	**99.99**	99.42
		MobileV2	84.85	99.40	85.15	82.24	97.40	67.34	**99.09**	92.65
		PARN101	89.41	99.60	81.39	59.74	**95.43**	80.03	90.97	85.25
		GoogLeNet	92.99	100.00	83.35	97.43	98.71	43.47	**99.95**	98.01
	FC	VGG-16	89.06	99.00	92.25	75.76	69.17	27.41	**96.72**	77.32
		RN-18	88.17	99.60	85.40	98.82	82.63	99.84	**99.98**	99.68
		MobileV2	86.50	99.90	90.43	71.58	83.08	58.40	**99.36**	98.31
		PARN101	89.65	99.80	83.75	74.19	85.17	32.93	**99.78**	97.99
		GoogLeNet	92.14	100.00	81.23	99.62	79.16	99.54	**99.99**	99.02
	Nashville	VGG-16	88.89	98.60	82.06	69.08	49.29	39.70	94.13	**95.30**
		RN-18	87.86	99.00	54.02	92.08	53.43	95.01	**98.85**	92.52
		MobileV2	85.24	99.40	74.62	97.24	49.03	88.94	**99.69**	98.87
		PARN101	89.31	98.60	86.91	89.08	49.50	81.46	**97.97**	94.52
		GoogLeNet	92.79	99.80	82.98	98.79	56.86	97.28	**99.87**	95.18
	SIG	VGG-16	89.22	95.60	**97.57**	44.36	78.26	39.42	92.22	87.92
		RN-18	86.71	99.80	99.22	99.90	81.89	99.96	**99.99**	99.95
		MobileV2	84.45	90.80	**98.10**	80.94	80.13	28.82	95.54	95.44
		PARN101	88.92	99.80	99.12	**99.64**	85.90	89.71	98.80	99.23
		GoogLeNet	92.80	94.20	93.09	97.17	82.23	36.62	**97.97**	97.92
	Smooth	VGG-16	88.11	96.20	**87.52**	47.68	65.59	28.99	86.37	78.00
		RN-18	87.45	95.20	79.36	95.60	70.13	**99.82**	97.78	86.50
		MobileV2	85.80	96.00	76.06	81.10	65.31	30.72	**93.08**	82.48
		PARN101	88.57	96.20	67.70	91.98	52.64	**93.01**	83.18	73.38
		GoogLeNet	92.75	96.00	87.03	94.18	38.93	**98.66**	95.05	89.91
	Trojan	VGG-16	88.47	98.20	88.42	47.16	88.90	47.80	**99.11**	99.00
		RN-18	87.63	98.40	81.71	98.74	91.88	93.76	**99.85**	98.81
		MobileV2	85.03	98.00	80.81	67.62	90.14	43.66	**98.22**	88.97
		PARN101	88.30	98.80	87.31	56.20	**94.12**	80.96	91.17	83.60
		GoogLeNet	92.94	98.20	91.54	78.24	94.21	29.71	**96.20**	85.44
	WaNet	VGG-16	88.53	97.80	58.73	44.48	90.24	53.94	**96.73**	92.21
		RN-18	87.51	98.40	51.42	91.41	80.45	**99.75**	95.71	85.04
		MobileV2	85.75	96.40	70.96	68.66	83.39	41.85	**99.06**	90.36
		PARN101	88.46	98.40	37.66	52.72	76.00	67.61	**87.34**	70.32
		GoogLeNet	92.70	99.80	64.52	97.51	94.79	94.81	**99.62**	95.05
GTSRB	BadNets	VGG-16	97.58	100.00	49.39	99.48	72.69	70.01	**100.00**	99.99
		RN-18	98.12	100.00	87.73	99.45	68.38	65.09	97.35	**100.00**
		MobileV2	97.32	100.00	64.27	99.43	58.63	81.98	**100.00**	98.71
		PARN101	96.68	100.00	39.94	92.95	39.14	55.06	**100.00**	99.73
		GoogLeNet	99.20	100.00	97.56	**99.74**	47.47	77.72	99.06	99.50
ImageNet Subset	BadNet	RN-18	58.13	100.00	94.05	97.87	98.94	99.83	**100.00**	**100.00**
		EfficientNet-b0	66.41	100.00	72.75	93.63	93.16	99.64	**100.00**	**100.00**
	ISSBA	RN-18	57.19	99.00	74.76	99.65	61.19	37.07	**99.72**	93.01
		EfficientNet-b0	62.03	100.00	61.85	99.93	65.69	59.91	**100.00**	96.21
Average	-	-	87.66	97.90	82.23	81.16	78.03	63.39	**96.11**	91.81



(a) BadNets (b) Blend (c) CL (d) DFST (e) Dynamic (f) FC

(g) Nashville (h) SIG (i) Smooth (j) Trojan (k) WaNet (l) ISSBA

Figure 14: The detection performance (AUC) on test set with different attacks under different poisoning rates [1%, 2.5%, 5%, 7.5%, 10%]. Results are averaged across different models.

Table 7: The detection performance on test set measured by AUC ((%)) for our CD method and the baselines against 12 backdoor attacks. The poisoning rate is 5% for all attacks. The Clean Accuracy (CA) is evaluated on clean test set. The best detection results are in **bold**.

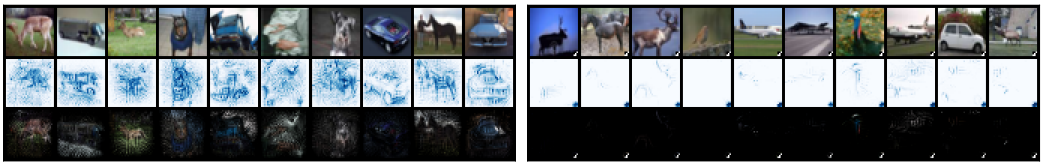
Dataset	Attack	Model	CA	ASR	AC	STRIP	SS	CD-L	CD-F
CIFAR10	BadNets	VGG-16	89.05	100.00	72.08	93.95	14.48	**100.00**	99.96
		RN-18	87.60	100.00	90.86	98.61	10.74	**100.00**	99.97
		MobileV2	84.19	100.00	91.52	98.15	72.02	**99.79**	94.70
		PARN101	89.77	100.00	36.43	**98.54**	97.85	77.63	71.90
		GoogLeNet	92.84	100.00	82.29	**99.05**	32.43	96.18	82.91
	Blend	VGG-16	89.26	99.60	59.55	74.25	18.34	96.53	**96.96**
		RN-18	87.77	100.00	97.34	78.02	58.43	**99.77**	96.17
		MobileV2	84.81	99.40	40.17	85.55	77.30	**97.98**	84.83
		PARN101	89.66	99.80	40.47	90.09	20.26	**97.69**	94.49
		GoogLeNet	92.39	99.20	**92.11**	87.18	28.30	75.22	89.63
	CL	VGG-16	89.00	98.40	18.78	93.52	44.32	93.08	**96.05**
		RN-18	87.33	98.40	29.54	96.95	22.26	**97.48**	91.70
		MobileV2	85.15	90.00	39.38	**98.14**	47.00	87.14	74.19
		PARN101	89.65	93.60	24.62	**97.20**	45.10	83.58	79.29
		GoogLeNet	92.69	53.60	16.09	**94.41**	36.39	65.27	60.33
	DFST	VGG-16	88.09	100.00	86.80	63.30	7.72	93.06	**95.15**
		RN-18	87.34	100.00	96.51	59.93	73.92	**97.22**	94.49
		MobileV2	84.67	99.80	**83.46**	59.90	24.04	82.25	69.12
		PARN101	89.37	99.60	69.32	57.72	16.17	80.81	**87.26**
		GoogLeNet	92.90	100.00	**98.78**	51.71	81.61	95.66	67.37
	Dynamic	VGG-16	88.94	100.00	60.53	77.44	33.08	**98.93**	98.68
		RN-18	87.93	100.00	93.78	82.74	49.19	**99.97**	99.35
		MobileV2	84.85	99.40	79.60	96.75	67.43	**98.54**	91.85
		PARN101	89.41	99.60	54.23	**93.72**	75.30	91.88	86.03
		GoogLeNet	92.99	100.00	97.20	98.12	29.53	**99.92**	97.89
	FC	VGG-16	89.06	99.00	87.52	70.44	83.47	**92.79**	82.23
		RN-18	88.17	99.60	94.93	83.07	58.01	**99.68**	99.08
		MobileV2	86.50	99.90	79.55	79.36	69.23	**99.26**	97.81
		PARN101	89.65	99.80	85.54	82.51	45.66	**99.40**	97.76
		GoogLeNet	92.14	100.00	96.14	69.49	66.74	**99.96**	98.71
	Nashville	VGG-16	88.89	98.60	50.02	45.40	24.16	92.05	**92.26**
		RN-18	87.86	99.00	87.50	53.26	69.54	**97.08**	91.20
		MobileV2	85.24	99.40	96.33	44.50	90.32	**99.59**	98.91
		PARN101	89.31	98.60	53.99	44.96	36.43	**98.04**	93.95
		GoogLeNet	92.79	99.80	92.71	53.37	82.63	**99.95**	94.97
	SIG	VGG-16	89.22	95.60	40.89	55.44	38.97	72.27	**76.74**
		RN-18	86.71	99.80	97.71	69.52	95.26	**99.85**	99.68
		MobileV2	84.45	90.80	43.10	47.35	43.47	91.42	**93.88**
		PARN101	88.92	99.80	48.44	67.77	49.20	96.41	**98.35**
		GoogLeNet	92.80	94.20	54.41	47.12	36.61	94.57	**97.21**
	Smooth	VGG-16	88.11	96.20	63.37	63.27	25.35	**82.23**	77.65
		RN-18	87.45	95.20	88.60	67.64	79.47	**94.30**	85.27
		MobileV2	85.80	96.00	78.83	62.87	25.76	**92.48**	81.56
		PARN101	88.57	96.20	61.75	48.10	32.85	**82.60**	75.77
		GoogLeNet	92.75	96.00	89.87	37.15	92.25	**93.52**	89.31
	Trojan	VGG-16	88.47	98.20	60.54	89.21	27.25	**98.15**	98.14
		RN-18	87.63	98.40	97.31	92.52	41.21	**99.17**	98.35
		MobileV2	85.03	98.00	71.84	90.85	44.33	**97.79**	89.50
		PARN101	88.30	98.80	58.61	**93.94**	76.13	92.83	86.57
		GoogLeNet	92.94	98.20	69.61	94.19	36.26	**95.65**	86.84
	WaNet	VGG-16	88.53	97.80	56.29	88.48	31.21	**95.63**	91.15
		RN-18	87.51	98.40	91.14	83.57	96.63	**97.09**	89.39
		MobileV2	85.75	96.40	60.15	79.14	47.02	**98.49**	88.55
		PARN101	88.46	98.40	43.49	77.36	33.32	**89.55**	77.55
		GoogLeNet	92.70	99.80	98.24	94.66	78.19	**99.65**	95.51
GTSRB	BadNets	VGG-16	97.58	100.00	59.50	76.42	77.76	**100.00**	100.00
		RN-18	98.12	100.00	83.77	70.27	76.22	96.98	**100.00**
		MobileV2	97.32	100.00	68.54	61.03	90.09	**100.00**	98.82
		PARN101	96.68	100.00	54.55	41.23	23.16	**100.00**	99.75
		GoogLeNet	99.20	100.00	97.61	49.02	97.05	98.72	**99.71**
ImageNet Subset	BadNets	RN-18	58.13	100.00	**100.00**	98.78	9.66	**100.00**	**100.00**
		EfficientNet-b0	66.41	100.00	**100.00**	92.90	8.74	**100.00**	**100.00**
	ISSBA	RN-18	57.19	99.00	73.13	57.50	59.16	**73.31**	67.89
		EfficientNet-b0	62.03	100.00	74.21	58.25	35.01	**74.45**	70.20
Average	-	-	87.66	97.90	71.27	75.11	50.27	**93.60**	90.01

Table 8: The detection performance on test set measured by true positive rate (TPR) and false positive rate (FPR) for our CD method and the baselines against 12 backdoor attacks, reported as TPR/FPR. For detection methods, classification threshold t is set to 1 standard deviation away from the mean calculated for 1% of the clean training set. The poisoning rate is 5% for all attacks. The best detection results are in **bold**, best is selected base on TPR first then use FPR for tie-breaking. TPR higher the better, FPR lower the better.

Dataset	Attack	Model	AC	STRIP	SS	CD-L	CD-F
CIFAR10	BadNets	VGG-16	99.60/68.14	87.60/11.18	19.40/68.04	100.00/10.97	**100.00/7.93**
		RN-18	99.80/51.98	**100.00/8.95**	93.80/79.08	100.00/14.54	100.00/14.33
		MobileV2	92.80/38.27	98.60/9.33	91.00/65.37	**99.80/15.52**	92.20/16.49
		PARN101	38.20/58.63	**100.00/9.20**	100.00/73.41	44.00/14.68	35.20/14.93
		GoogLeNet	99.40/53.12	**100.00/8.57**	30.20/75.32	94.60/15.48	65.40/16.15
	Blend	VGG-16	**94.20/62.82**	44.80/12.39	41.60/66.66	92.80/12.26	93.60/9.94
		RN-18	98.80/59.66	50.60/12.09	**100.00/80.13**	99.80/14.46	94.80/14.64
		MobileV2	20.00/35.73	66.00/11.71	97.00/64.02	**97.40/16.32**	69.00/16.76
		PARN101	51.40/52.78	77.00/10.59	46.60/73.94	**97.40/14.71**	91.80/15.17
		GoogLeNet	**93.80/50.84**	71.80/11.66	46.40/67.93	37.60/15.53	81.00/16.05
	CL	VGG-16	4.80/60.75	86.20/11.76	54.00/68.49	84.80/12.62	**89.80/9.81**
		RN-18	14.40/52.54	**97.00/9.79**	56.40/77.96	95.20/13.59	81.40/13.71
		MobileV2	16.00/36.38	**97.00/9.97**	67.00/66.81	72.60/16.82	46.80/17.07
		PARN101	5.80/54.40	**96.40/9.53**	64.20/72.77	52.20/13.36	47.60/13.99
		GoogLeNet	0.00/43.00	**88.40/10.48**	65.20/70.57	28.40/14.89	22.80/14.75
	DFST	VGG-16	**99.40/66.74**	10.80/14.66	0.60/55.87	82.60/11.73	85.80/10.75
		RN-18	97.20/58.31	8.40/13.60	**100.00/80.61**	95.80/14.20	91.20/13.96
		MobileV2	**89.00/36.46**	4.40/15.31	28.80/62.75	66.40/16.18	37.80/16.93
		PARN101	**92.80/54.97**	8.40/13.77	44.40/72.36	50.60/14.62	72.40/14.87
		GoogLeNet	99.00/53.41	6.40/14.81	**100.00/75.08**	92.80/15.24	24.60/15.44
	Dynamic	VGG-16	86.80/62.71	61.40/11.49	79.80/70.18	**97.60/13.38**	97.20/10.74
		RN-18	96.20/58.81	62.40/10.20	99.20/79.74	**100.00/13.91**	99.40/14.09
		MobileV2	78.40/38.21	91.80/7.65	84.60/65.57	**97.40/15.51**	84.60/16.23
		PARN101	60.00/53.23	83.20/9.20	**89.60/73.20**	80.60/13.61	66.20/14.71
		GoogLeNet	99.00/53.68	94.80/7.74	59.60/71.93	**100.00/15.66**	96.80/15.82
	FC	VGG-16	**99.80/90.64**	56.71/22.62	90.38/85.27	96.09/35.46	82.87/33.40
		RN-18	99.60/99.03	72.75/17.68	92.99/81.24	**99.80/34.84**	99.10/27.91
		MobileV2	96.79/90.31	66.23/16.70	55.11/81.99	**99.70/33.07**	98.30/30.31
		PARN101	**99.90/93.91**	71.94/17.45	95.49/74.02	99.70/33.34	98.60/29.83
		GoogLeNet	99.70/99.05	53.51/19.48	97.90/73.22	**100.00/36.57**	99.10/31.09
	Nashville	VGG-16	82.80/60.27	9.40/13.61	76.80/71.66	83.60/13.48	**84.00/10.00**
		RN-18	91.00/59.96	12.20/13.27	**98.20/80.32**	93.40/14.55	82.00/14.52
		MobileV2	97.60/35.34	8.60/13.96	**99.60/65.47**	99.40/15.45	98.60/16.27
		PARN101	67.40/50.68	9.40/14.06	96.00/74.09	**96.60/14.55**	88.80/14.69
		GoogLeNet	98.40/44.71	11.80/14.85	100.00/71.71	**100.00/16.05**	89.40/15.75
	SIG	VGG-16	**69.60/63.18**	23.20/13.35	43.20/63.29	39.00/12.20	40.00/8.61
		RN-18	98.20/53.76	37.00/12.46	**100.00/80.44**	99.80/12.45	99.80/13.72
		MobileV2	14.20/37.92	12.00/14.71	71.60/67.08	83.20/15.64	**91.40/16.76**
		PARN101	93.40/61.20	32.60/12.03	90.40/62.49	93.40/12.99	**98.60/14.17**
		GoogLeNet	51.00/43.82	15.80/13.74	88.20/72.17	88.20/14.95	**96.40/15.61**
	Smooth	VGG-16	**83.20/54.71**	36.80/12.36	34.40/67.78	63.60/13.22	28.00/7.89
		RN-18	91.00/52.63	41.80/11.60	**100.00/80.20**	88.00/14.54	67.60/14.58
		MobileV2	78.40/38.07	35.00/13.16	43.60/70.06	**86.20/16.12**	60.60/17.11
		PARN101	**80.80/58.97**	18.80/12.71	51.20/70.69	61.00/13.99	40.40/14.54
		GoogLeNet	90.80/51.97	10.80/14.67	**99.60/73.55**	88.00/15.51	77.40/15.85
	Trojan	VGG-16	94.80/62.41	75.80/10.99	70.60/69.80	**97.00/12.78**	96.80/8.48
		RN-18	98.00/61.13	83.60/9.80	**99.60/79.44**	98.40/13.66	97.40/14.40
		MobileV2	71.60/36.98	78.20/10.20	57.20/64.95	**96.80/15.35**	82.60/16.67
		PARN101	88.20/57.54	87.00/9.28	**95.40/74.43**	84.60/14.45	68.60/15.14
		GoogLeNet	91.20/55.22	86.40/8.99	50.60/70.56	**92.40/15.02**	72.40/15.64
	WaNet	VGG-16	81.60/58.32	68.00/11.17	81.00/70.65	**92.80/12.17**	81.60/11.13
		RN-18	93.60/56.32	55.00/11.93	**100.00/80.69**	99.40/10.73	76.60/13.82
		MobileV2	54.40/36.92	43.00/11.80	92.80/68.62	**98.20/15.09**	78.00/16.17
		PARN101	51.80/53.60	46.00/12.06	69.60/74.51	**78.60/15.13**	44.80/13.51
		GoogLeNet	98.40/52.62	87.40/10.51	**100.00/74.91**	99.80/14.82	93.00/16.26
GTSRB	BadNets	VGG-16	95.88/65.53	33.44/10.15	100.00/56.86	**100.00/4.88**	86.21/0.00
		RN-18	100.00/92.02	29.16/12.63	100.00/68.26	100.00/11.06	**100.00/0.54**
		MobileV2	99.21/72.83	25.36/18.39	100.00/61.14	**100.00/2.06**	79.56/0.00
		PARN101	98.57/78.88	2.22/15.74	90.17/75.54	**100.00/7.85**	98.89/2.90
		GoogLeNet	100.00/81.97	14.10/16.13	100.00/79.63	**100.00/9.68**	27.26/0.06
ImageNet Subset	BadNets	RN-18	100.00/53.11	99.80/13.35	61.00/71.81	100.00/32.99	**100.00/22.15**
		EfficientNet-b0	**100.00/10.61**	90.00/15.49	0.40/70.23	100.00/20.07	100.00/20.27
	ISSBA	RN-18	**100.00/76.18**	35.00/28.26	66.00/45.83	98.00/78.44	90.00/61.19
		EfficientNet-b0	100.00/60.33	43.00/31.68	6.00/26.88	**100.00/50.92**	89.00/49.67
Average	-	-	80.12/57.47	53.32/13.10	73.82/70.77	**88.36/17.28**	78.92/15.87

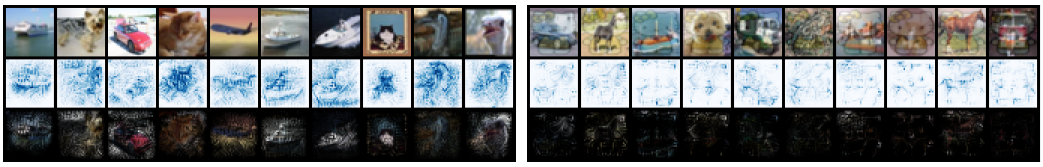
D ADDITIONAL VISUALIZATIONS

Figures 15 to 28 show the input images, learned mask, and distilled CP of both clean and backdoored samples for each corresponding experiment evaluated in section 4. The masks and CP are extracted from the corresponding backdoored model (ResNet-18) using CD-L.



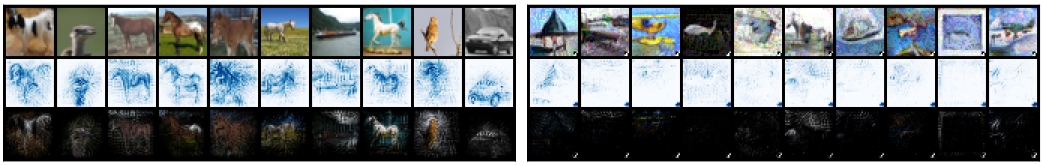
(a) Clean samples (b) Backdoor samples

Figure 15: Corresponding input images, learned masks and CPs for BadNets (Gu et al., 2017) on CIFAR10.



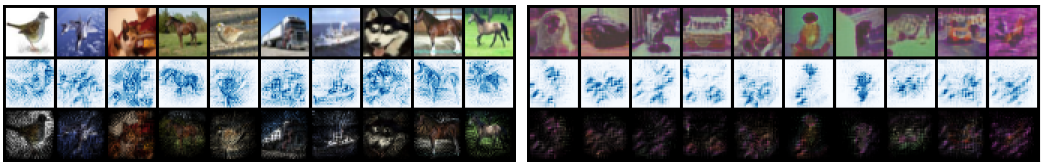
(a) Clean samples (b) Backdoor samples

Figure 16: Corresponding input images, learned masks and CPs for Blend (Chen et al., 2017) on CIFAR10.



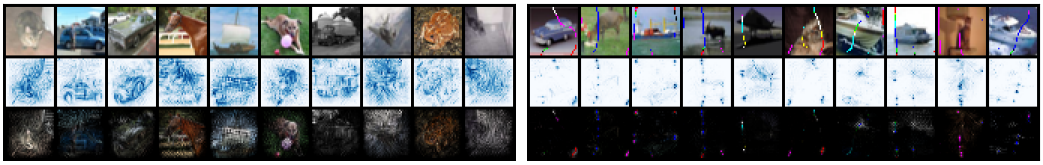
(a) Clean samples (b) Backdoor samples

Figure 17: Corresponding input images, learned masks and CPs for CL (Turner et al., 2018) on CIFAR10.



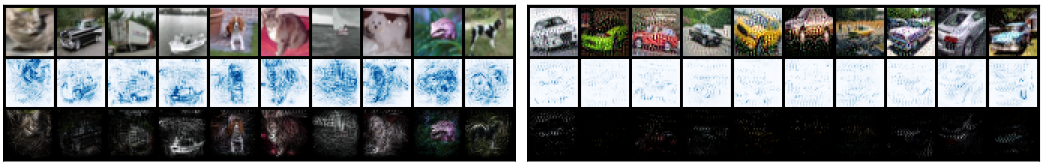
(a) Clean samples (b) Backdoor samples

Figure 18: Corresponding input images, learned masks and CPs for DFST (Cheng et al., 2020) on CIFAR10.



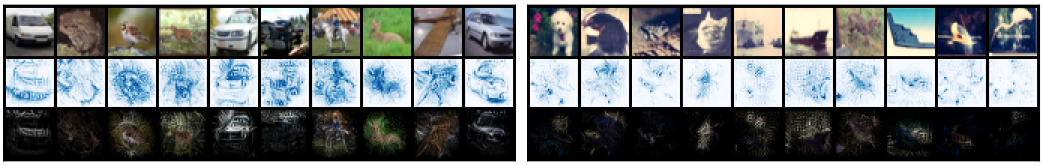
(a) Clean samples (b) Backdoor samples

Figure 19: Corresponding input images, learned masks and CPs for Dynamic (Nguyen & Tran, 2020) on CIFAR10.



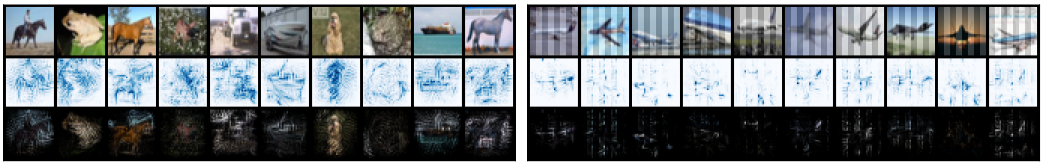
(a) Clean samples (b) Backdoor samples

Figure 20: Corresponding input images, learned masks and CPs for FC (Shafahi et al., 2018) on CIFAR10.



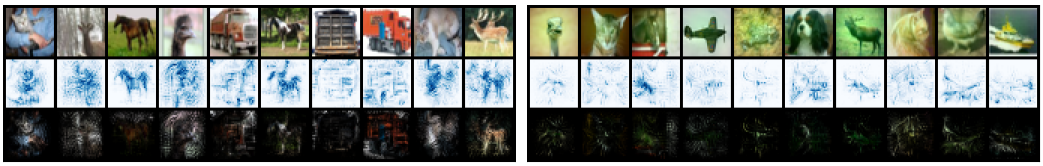
(a) Clean samples (b) Backdoor samples

Figure 21: Corresponding input images, learned masks and CPs for Nashville (Liu et al., 2019) on CIFAR10.



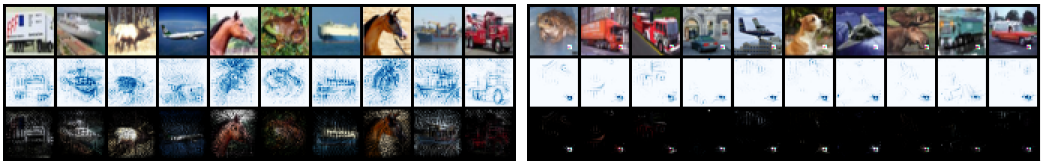
(a) Clean samples (b) Backdoor samples

Figure 22: Corresponding input images, learned masks and CPs for SIG (Barni et al., 2019) on CIFAR10.



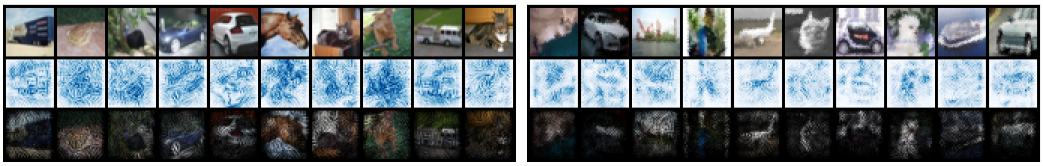
(a) Clean samples (b) Backdoor samples

Figure 23: Corresponding input images, learned masks and CPs for Smooth (Zeng et al., 2021) on CIFAR10.



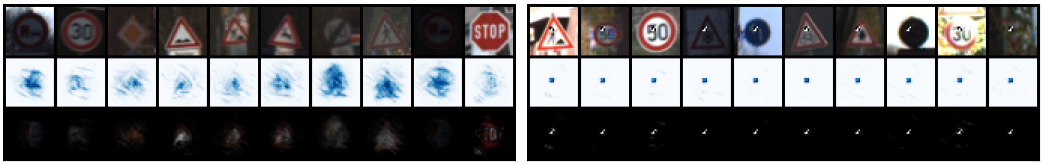
(a) Clean samples (b) Backdoor samples

Figure 24: Corresponding input images, learned masks and CPs for Trojan (Liu et al., 2018b) on CIFAR10.



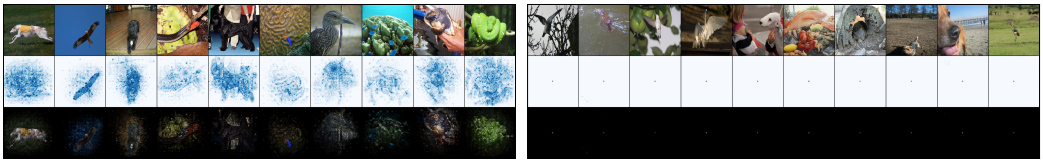
(a) Clean samples (b) Backdoor samples

Figure 25: Corresponding input images, learned masks and CPs for using WaNet trigger (Nguyen & Tran, 2021) on CIFAR10.



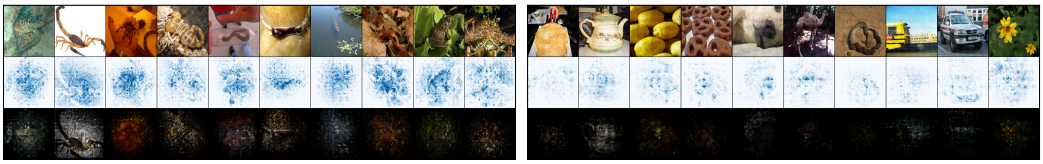
(a) Clean samples (b) Backdoor samples

Figure 26: Corresponding input images, learned masks and CPs for BadNets (Gu et al., 2017) on GTSRB.



(a) Clean samples (b) Backdoor samples

Figure 27: Corresponding input images, learned masks and CPs for BadNets (Gu et al., 2017) on ImageNet subset.



(a) Clean samples (b) Backdoor samples

Figure 28: Corresponding input images, learned masks and CPs for ISSBA (Li et al., 2021c) on ImageNet subset.