Appendix

A. Formal Analysis on the Transformation Layer

As a region of the feature representation is transformed by a convolutional kernel $\boldsymbol{U}$, we study the property of such an operation for expressing backdoor behaviors. Assume a 2×2 input region $\boldsymbol{X}$ on the left of Figure 5 and a kernel parameterized by $\boldsymbol{W} \in \mathbb{R}^{2\times2}$. Zero-padding is used (demonstrated by the dotted cells). Output values can be derived from the values in the region and the parameter values through the following equations.

$$
\begin{aligned}
a_0 &= w_0 \cdot x_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3 \\
a_1 &= w_0 \cdot x_1 + w_2 \cdot x_3 \\
a_2 &= w_0 \cdot x_2 + w_1 \cdot x_3 \\
a_3 &= w_0 \cdot x_3
\end{aligned} \tag{9}
$$

Here, we leave the activation functions out for discussion simplicity. Suppose a backdoor applies adversarial perturbation $\boldsymbol{\delta}$ on the region $\boldsymbol{X}$. That is, $x'_i = x_i + \delta_i, i \in \{0, 1, 2, 3\}$. The feature representation for the backdoor sample region $\boldsymbol{A'}$ is hence the following.

$$
\begin{aligned}
a'_0 &= w_0 \cdot (x_0 + \delta_0) + w_1 \cdot (x_1 + \delta_1) + w_2 \cdot (x_2 + \delta_2) \\
&\quad + w_3 \cdot (x_3 + \delta_3) \\
a'_1 &= w_0 \cdot (x_1 + \delta_1) + w_2 \cdot (x_3 + \delta_3) \\
a'_2 &= w_0 \cdot (x_2 + \delta_2) + w_1 \cdot (x_3 + \delta_3) \\
a'_3 &= w_0 \cdot (x_3 + \delta_3)
\end{aligned} \tag{10}
$$

Our goal is to derive backdoor samples from benign inputs. That is, we apply the convolutional operation on the benign feature representation to produce the backdoor representation. Here, we use a convolutional kernel $\boldsymbol{U} \in \mathbb{R}^{2\times2}$ for analysis simplicity. Applying the kernel on the normal representation $\boldsymbol{A}$ (see the middle part of Figure 5) produces the following.

$$
\begin{aligned}
\hat{a}_0 &= u_0 \cdot a_0 + u_1 \cdot a_1 + u_2 \cdot a_2 + u_3 \cdot a_3 \\
&= u_0 \cdot (w_0 x_0 + w_1 x_1 + w_2 x_2 + w_3 x_3) \\
&\quad + u_1 \cdot (w_0 x_1 + w_2 x_3) + u_2 \cdot (w_0 x_2 + w_1 x_3) \\
&\quad + u_3 \cdot w_0 x_3 \\
\hat{a}_1 &= u_0 \cdot a_1 + u_2 \cdot a_3 \\
&= u_0 \cdot (w_0 x_1 + w_2 x_3) + u_2 \cdot w_0 x_3 \\
\hat{a}_2 &= u_0 \cdot a_2 + u_1 \cdot a_3 = u_0 \cdot (w_0 x_2 + w_1 x_3) + u_1 \cdot w_0 x_3 \\
\hat{a}_3 &= u_0 \cdot a_3 = u_0 \cdot w_0 x_3
\end{aligned} \tag{11}
$$

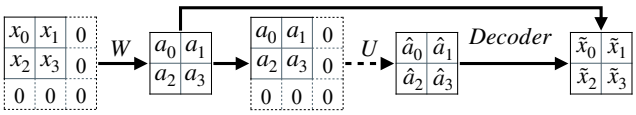

Figure 5. Example for transforming internal values.

Let $\boldsymbol{A'} = \hat{\boldsymbol{A}}$ and we have

$$
\begin{aligned}
\delta_0 &= (u_0 - 1) \cdot x_0 + u_1 \cdot x_1 + u_2 \cdot x_2 + u_3 \cdot x_3 \\
\delta_1 &= (u_0 - 1) \cdot x_1 + u_2 \cdot x_3 \\
\delta_2 &= (u_0 - 1) \cdot x_2 + u_1 \cdot x_3 \\
\delta_3 &= (u_0 - 1) \cdot x_3
\end{aligned} \tag{12}
$$

As observed in Figure 4, semantic backdoors transform inputs based on each original pixel value and do not introduce abrupt value changes in the neighborhood of each pixel (within the region)[3]. That is, each pixel perturbation introduced by the backdoor transformation correlates to the original value of its corresponding pixel and the neighboring pixels. This can be expressed by our method as show in Equation 12. For instance, the perturbation on the first pixel δ_0 is a portion $(u_0 - 1)$ of the corresponding pixel x_0 and also the linear combination of neighboring pixels $(u_1 x_1 + u_2 x_2 + u_3 x_3)$. The scale of the perturbation is parameterized by our convolutional transformation $\boldsymbol{U}$. It can be properly modeled during our backdoor generation using the gradient information from the victim model. Note that although δ_1-δ_3 may not involve some neighboring pixels, that is because we have only one layer. In practice, a model has many layers, and x_0-x_3 are feature values from the previous layer, which are functions involving neighboring pixels. In addition, the above analysis only considers one convolutional kernel in our transformation layer within the region for discussion simplicity. In practice, for example, the feature representation has 64 channels and each channel is associated with one kernel, which gives us 64 different combinations of neighboring pixels for each region.

B. Details of Experiment Setup

Datasets and Models.

- **CIFAR-10** [24] is an object recognition dataset with 10 classification classes. It consists of 60,000 images and is divided into a training set (48,000 images), a validation set (2,000 images), and a test set (10,000 images).
- **STL-10** [11] is an image recognition dataset with 10 classification classes. It consists of 5,000 training images and 8,000 test images.
- **SVHN** [40] is a dataset contains house digital numbers extracted from Google Street View images. It has 73,257 training images and 26,032 test images. We divide the

[3] Static backdoors, such as BadNets, introduce strong features distinct from benign features. Our formulation can capture the introduced features, such as the color scheme. These features can be easily mitigated by `MARTINI`, as shown by our experiments in Section 5.1.

original training set into 67,257 images for training and 6,000 images for validation.

- **GTSRB** [51] is a German traffic sign recognition dataset with 43 classes. We split the dataset into a training set (35,289 images), a validation set (3,920 images), and a test set (12,630 images).
- **CelebA** [37] is a face attributes dataset. It contains 10,177 identities with 202,599 face images. Each image has an annotation of 40 binary attributes. We follow [41] to select 3 out of 40 attributes, i.e., Heavy Makeup, Mouth Slightly Open, and Smiling, and create an 8-class classification task.
- **TrojAI** [43] round 4 includes 16 types of model structures such as InceptionV3 [52], DenseNet121 [19], SqueezeNet [21], etc. The task of these models is to recognize synthetic street traffic signs with between 15 and 45 classes. Input images are constructed by compositing a foreground object, e.g., a synthetic traffic sign, with a random background images from five different dataset such as Cityscapes [12], KITTI [17], Swedish Roads [25], etc. A set of random transformations are applied during model training, such as blurring, lighting, shifting, titling, etc. Adversarial training such as PGD [38] and FBF [64] is also utilized to improve model quality. We randomly select 34 poisoned models by filter attack from TrojAI round 4 [43].

Baseline Mitigation Methods.

- **Fine-tuning (FT)** [27] is a standard method originally proposed for transfer learning. It updates a pre-trained model's weights with a small learning rate on the training set. We leverage the finetuning baseline setting in NAD [27], which adopts data augmentation techniques including random crop, horizontal flipping, and cutout [13] during training.
- **Fine-pruning (FP)** [32] prunes neurons that have low activation values for a set of clean samples. It then finetune the pruned model on a small set of clean samples.
- **MCR** [70] linearly interpolates the weight parameters of two models. It also includes a set of trainable parameters during the interpolation. Specifically, the following equation is used to build a new model $\phi_\theta(t)$.

$$\phi_\theta(t) = (1-t)^2\omega_1 + 2t(1-t)\theta + t^2\omega_2, \quad 0 \leq t \leq 1, \quad (13)$$

where t is the interpolation hyper-parameter ranging from 0 to 1. ω_1 and ω_2 are the weight parameters of two pre-trained models, which are fixed. θ is a set of trainable parameters that have the same shape of ω_1 and ω_2. For eliminating backdoors in poisoned models, MCR uses the poisoned model and its finetuned version as the two endpoints (ω_1 and ω_2) and trains θ on a small set of clean samples. The best t is chosen for the interpolation based on the clean accuracy.

- **NAD** [27] leverages the teacher-student structure to eliminate backdoors. It first finetunes the poisoned model on 5% of the training set. It uses this finetuned model as the teacher network, and the poisoned model as the student network. It then aims to reduce the internal feature differences between the teacher network and the student network by updating the student network. Finally, NAD outputs the student network as the cleaned model.
- **ANP** [65] is based on the observation that backdoor related neurons are more sensitive to adversarial perturbations on their weights. It hence applies a mask on all the neurons in the model, adversarially perturbs neuron weights to increase the classification loss for a set of clean samples, and minimizes the size of mask. ANP then prunes neurons with small mask values, meaning that they have been compromised by backdoor attacks.
- **ABS** [35] introduces a neuron stimulation analysis to expose abnormal behaviors of neurons in a deep neural network by increasing their activation values. Those neurons are regarded as compromised neurons and leveraged to reverse engineer backdoor triggers. ABS proposes a one-layer transformation to approximate/invert filter triggers. The inverted trigger is hence utilized to remove the injected backdoor in poisoned models following the unlearning procedure in NC [61].
- **MOTH** [54] enhances model robustness by increasing the distance between classes. It employs trigger inversion techniques to generate adversarial samples that bridge class separations and utilizes asymmetric training to harden the model. MOTH mitigates backdoor effects by disrupting the shortcut connection between victim classes and the target class.
- **I-BAU** [68] introduces a minimax formulation to mitigate the backdoor effect. Specifically, this method leverages the implicit hypergradient to address the interdependence between trigger synthesis and adversarial training processes.
- **SEAM** [72] leverages the phenomenon of catastrophic forgetting to unlearn the backdoor effect through label shuffling. It then seeks to restore clean knowledge by fine-tuning with the correct labels. This method effectively disrupts the connection between the backdoor trigger and its target label.
- **FT-SAM** [71] represents a novel backdoor defense paradigm that integrates sharpness-aware minimization with fine-tuning. This approach specifically targets neurons associated with the backdoor, aiming to reduce their influence by shrinking their norms, thereby mitigating the backdoor effect.
- **FST** [39] proposes a simple yet effective technique for purifying backdoors through finetuning. It specifically promotes shifts in feature representation by actively diverging the classifier weights from their initially

compromised states.

- **RNP** [30] exposes and eliminates backdoor neurons through a process of unlearning followed by recovery. Specifically, RNP begins by maximizing the model's error using a small subset of clean data. Afterward, it recovers the affected neurons by minimizing the model's error on the same dataset. Neurons that remain problematic after this process are considered backdoored and are subsequently pruned.

Evaluated Backdoor Attacks.

- **BadNets** [18] is the pioneering study that first highlighted backdoor threats in deep learning models. It employs a static patch, e.g., a flower placed in the corner of an image, as the backdoor trigger. Any input containing this patch is then misclassified to the attack target label.
- **Dynamic attack** [47] utilizes a generative model to create dynamic patch triggers that vary in pattern and location across images. This diversity enhances its stealthiness against backdoor detection methods.
- **Input-aware attack (IA)** [42] improves upon dynamic backdoors by generating more diverse and sample-specific triggers to evade backdoor detection. The IA triggers are more invisible in the input space.
- **Deep Feature Space Trojan (DFST)** [7] leverages a generative adversarial network (GAN) to inject a certain style (e.g., sunrise color style) to given training samples. It also introduces a detoxification procedure by iteratively training on ABS [35] reverse-engineered backdoors to reduce the number of compromised neurons that can be leveraged by existing scanners for successful detection. We follow the original paper and poison models with two settings: one-round detoxification and three-rounds detoxification.
- **Blend attack** [6] injects a random perturbation pattern on the training samples of non-target classes and changes the ground truth labels of these samples to the target class (label 0). We use the random pattern reported in the original paper and use a blend ratio of $\alpha = 0.2$.
- **Adaptive Blend (A-Blend)** [45] refines the typical blend attack by including correctly labeled trigger-planted samples to enhance backdoor learning regularization. It also introduces asymmetric trigger strategies that improve the ASR and diversify the representations of poisoned samples.
- **Sinusoidal Signal attck (SIG)** [2] injects a strip-like pattern on the training samples of the target class and retains the original ground truth labels. We follow the setting in the original paper and generate the backdoor pattern using the horizontal sinusoidal function with $\Delta = 20$ and $f = 6$. We use label 0 as the target class and poison 8% of the training data in the target class.
- **LIRA** [14] designs a learnable trigger injection function to be used during model poisoning. Specifically, it trains a generative model to inject triggers concurrently with backdoor model training. LIRA utilizes the dataset itself to enhance the specificity of the backdoor triggers.
- **WaNet** [41] uses elastic image warping that deforms an image by applying the distortion transformation (e.g., distorting straight lines) as the backdoor. We download three backdoored models from the official repository [41], which are trained on CIFAR-10, GTSRB, and CelebA, respectively.
- **Invisible attack** [29] leverages a generator to encode a string (e.g., the index of a target label) onto an input image. We download the pre-trained generator from the official repository [28] and use it to inject invisible backdoors following the setting in the original paper.
- **Clean Label attack (CL)** [59] generates adversarial perturbations on the training samples in the target class using an adversarailly trained model. It then injects a 2×2 grid at the top left corner of the target-class inputs and retain their ground truth labels. We use L^∞ bound of $8/255$ for crafting adversarial perturbations, use label 3 as the target class, and poison 50% of the training data in the target class following the official repository [60].
- **Narcissus** [69] introduces a clean-label backdoor attack that is both stealthy and robust. Specifically, it trains a surrogate model to capture the important features from the target label, which are then used as the backdoor trigger. It selectively poisons only the images of the target class with this trigger, compelling the model to associate the trigger with the target label without altering the labels.
- **COMBAT** [20] improves the clean-label backdoor technique beyond Narcissus by leveraging a generative model to produce the triggers. It also incorporates frequency features and introduces an alternative training method to enhace the learning of the backdoor trigger function and the poisoned model.
- **Filter attack** [35] applies Instagram filters on training samples and changes the ground truth labels of these samples to the target class. There are various filters can be used to poison data, such as Gotham filter, Nashville filter, Kelvin filter, Lomo filter, Toaster filter, etc.

C. Results on Filter Attack

For filter attack, we leverage pre-trained backdoored models downloaded from the TrojAI competition [43]. The results are reported in Figure 6. The x-axis and y-axis denote the model IDs and the ASR, respectively. The bars in the light/dark colors show the ASR of injected backdoors before/after applying different defense techniques. Observe that FT (blue bars) can only repair half of the evaluated models (17 out of the 34 models). This is expected as backdoor attacks include clean data together with backdoored samples during training. Fine-tuning only on clean samples may not eliminate the backdoor patterns that have already

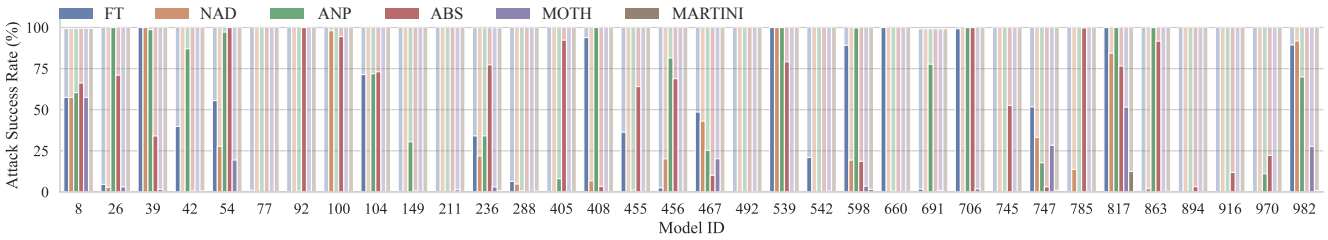

Figure 6. ASR of filter attack before (light color) and after (dark color) applying different defenses.

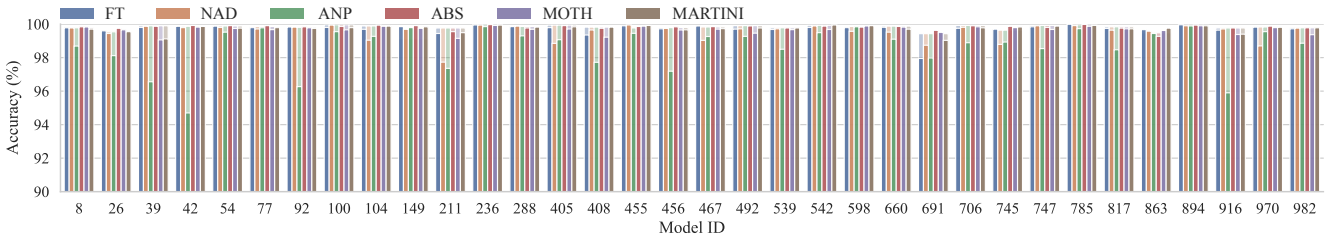

Figure 7. Acc. of filter-backdoored models before (light color) and after (dark color) applying different defenses.

been learned by backdoored models. NAD leverages the teacher-student structure and treats the model from FT as the teacher network. Its performance hence is limited by Fine-tuning. This can be observed from the orange bars in Figure 6. NAD is only able to eliminate five more backdoors (with a total of 21 models). ANP has limited performance on TrojAI models, with only 15 poisoned models being repaired. The TrojAI backdoored models were trained by NIST [43], and different training strategies including random transformations, adversarial training, etc., were employed to make injected backdoors more robust and hard to detect. These strategies may reduce the sensitivity of individual neurons on backdoor patterns. ANP is hence not able to identify compromised neurons and fails to remove injected backdoors. This observation is consistent with the results on DFST backdoors that apply detoxification to reduce compromised neurons.

ABS can only repair 15 models. As the injected backdoors in TrojAI models are label-specific, ABS may not be able to identify the correct victim-target class pair. The inverted triggers fail to expose the injected backdoor behaviors. Unlearning on those triggers hence cannot repair models. MOTH can eliminate more backdoors than other baselines with 28 fixed models. As discussed in the motivation section, semantic backdoors perturb all pixels on the input and are dynamic, while MOTH focuses on patch-like static backdoors. It can raise the bar for semantic backdoors to some extent but still fails to repair 6 TrojAI backdoored models. `MARTINI`, on the other hand, can eliminate all the backdoors with an average ASR down to 0.55%, outperforming the others. The accuracy of backdoored models before and after repair is shown in Figure 7 (in Appendix). Overall, all the approaches incur a very small accuracy degradation on average ($< 0.3\%$), except for ANP (1.16%).

Table 4. Adaptive Attacks.

Target	20% Poisoning		`MARTINI`		50% Poisoning		`MARTINI`	
	Acc.	ASR	Acc.	ASR	Acc.	ASR	Acc.	ASR
0	83.64%	60.43%	83.94%	9.94%	67.86%	74.94%	75.71%	4.42%
1	81.61%	58.48%	84.57%	10.46%	70.64%	79.99%	81.79%	6.01%
2	84.43%	56.88%	83.67%	12.79%	67.99%	77.74%	81.50%	5.70%
3	84.91%	60.22%	85.56%	7.18%	70.73%	74.81%	75.02%	0.79%
4	82.91%	76.90%	83.29%	10.55%	71.87%	75.35%	75.80%	9.47%
5	80.11%	76.04%	84.87%	5.00%	66.70%	78.09%	79.12%	11.28%
6	84.47%	64.13%	82.78%	12.93%	71.93%	73.24%	84.72%	14.31%
7	82.32%	76.69%	83.71%	13.11%	68.37%	84.39%	78.61%	11.09%
8	84.18%	62.56%	86.06%	12.16%	71.50%	75.27%	82.92%	12.86%
9	82.57%	69.63%	84.63%	9.14%	70.85%	75.50%	79.20%	10.29%

`MARTINI` has the smallest accuracy degradation of 0.06%.

D. Adaptive Attacks

We conduct an adaptive attack by optimizing a trigger pattern during poisoning while applying our mitigation method (the adaptive knowledge). The goal is to prevent the model from learning simple triggers that can be easily generated, making the final injected backdoor hard to invert and capable of evading our defense. The adaptive attack starts with a random trigger and stamps it on training images along with the target label for poisoning. At each training iteration, it also applies the inverted trigger, stamps it on images, and uses the ground truth label for training. The attack then optimizes the injected trigger on the current adversarially-trained model and uses this optimized trigger for injection. The poisoning process is iterative and continues until convergence.

The experiment is conducted on a ResNet20 model with CIFAR-10, evaluating different choices of the 10 classes as the target label. The results are shown in Table 4. The first column shows the target label. Columns 2-5 show the accuracy and ASR for the backdoored models (with a 20% poi-

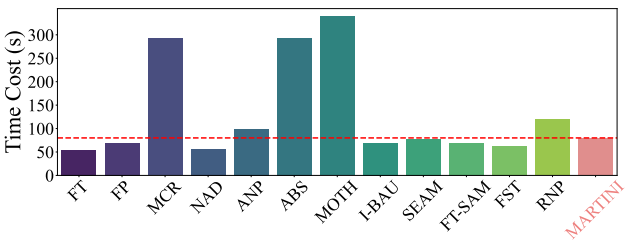

Figure 8. Time Cost.

soning rate) before and after repair by our method. Columns 6-9 present the results using a 50% poisoning rate for adaptive attacks. Observe that with a 20% poisoning rate, the backdoored models have an average accuracy of 83.12% and an ASR of 66.20%. By increasing the poisoning rate to 50%, the ASR improves to 76.93%, with a significant accuracy degradation to 69.84% on average. The ASRs are slightly higher for target labels 4, 5, and 7 with the 20% poisoning rate, and for label 7 with the 50% poisoning rate. As MARTINI aims to mitigate backdoors while the poisoning tries to inject a backdoor, these two contradicting goals result in the accuracy being much lower than a clean model (91.52%) and the ASR being relatively low as well. By applying our method to the poisoned models, the ASR drops to 10.33% (20% poisoning rate) and 8.62% (50% poisoning rate) without accuracy degradation (84.31% and 79.44% on average, respectively), as shown in Table 4. This delineates the resilience of our mitigation technique to adaptive attacks. Regarding different target labels, the ASRs for repaired models are slightly higher for labels 2, 6, and 7 with the 20% poisoning rate, and for labels 6 and 8 with the 50% poisoning rate. These slight variations are due to the fact that our method does not mitigate backdoors equally for all classes. Nonetheless, the repaired models all have less than 15% ASR, demonstrating the robustness of our technique against adaptive attacks with different target choices.

We also conduct an adaptive attack where backdoors have the same feature in different regions to counter our regional transformation. The results show that our method reduces the ASR from 98.80% to 0.28% with only a 0.83% accuracy degradation.

E. Defense Efficiency

We use an off-the-shelf encoder and only train the decoder, which takes 32.76 minutes. This is a one-time effort, and the trained decoder can be used for generating backdoors on different datasets.

Figure 8 shows the time in seconds for mitigating backdoors by different defenses. Most of the defense techniques can finish within 100 seconds. MCR, ABS, and MOTH have higher time costs, requiring more than 250 seconds. Overall, the time cost of MARTINI is comparable to that of other baselines. Recall that our method achieves more than 20% ASR reduction compared to baselines on

(a) Clean (b) Misclassification (c) Injection (d) Localization

Figure 9. Object detection backdoors.

Table 5. Mitigating backdoors in object detection. The best results highlighted with blue color.

Attack	Model	Original		FT		NAD		Ours	
		mAP	ASR	mAP	ASR	mAP	ASR	mAP	ASR
Miscls.	SSD	87.74%	100.00%	81.85%	92.38%	80.60%	75.24%	82.82%	9.05%
	F-RCNN	96.50%	100.00%	92.13%	71.43%	91.23%	15.24%	92.95%	5.71%
Inject.	SSD	85.20%	100.00%	79.79%	100.00%	79.29%	99.38%	81.23%	11.90%
	F-RCNN	97.03%	100.00%	90.62%	43.33%	89.82%	17.50%	91.16%	7.62%
Local.	SSD	85.32%	82.38%	82.61%	0.00%	82.02%	0.00%	85.21%	0.48%
	F-RCNN	96.91%	100.00%	91.84%	0.00%	91.74%	0.00%	92.50%	0.00%
Average		**91.45%**	**97.06%**	**86.47%**	**51.19%**	**85.78%**	**34.56%**	**87.65%**	**5.79%**

many attacks, especially on recent advanced attacks, which is a critical aspect of backdoor defense.

F. Extension to Other Applications

Object Detection. We leverage the TrojAI [43] dataset for object detection. This dataset consists of images synthesized with real street backgrounds and multiple traffic signs as foreground objects. We consider three types of backdoor attacks [4, 5, 31]: misclassification, injection, and localization. Figure 9(a) shows the clean input and its correct prediction. Figure 9(b) demonstrates the object misclassification attack, where a yellow triangle, serving as the backdoor trigger, is stamped on the stop sign, causing the model to mis-recognize the sign as the speed limit. Figure 9(c) illustrates the injection attack, where the model falsely recognizes the trigger as a stop sign. Figure 9(d) visualizes the object localization attack, where the trigger causes the predicted object bounding box to shift away from its correct location. We conduct experiments using two well-known model architectures, SSD [33] and Faster-RCNN (F-RCNN) [46]. The clean object detection performance is measured by mean average precision (mAP). We adapte two baselines from image classification, i.e., FT and NAD. The experimental results are presented in Table 5. On average, MARTINI reduces the ASR from 97% to below 6%, whereas the baselines still maintain ASR at high levels, over 34%, with greater performance sacrifices than MARTINI. This underscores the efficacy of MARTINI in mitigating backdoors in object detection, as it effectively approximates and eliminates the backdoor effects. Our mitigation is not perfect. In some cases, MARTINI still has a non-trivial ASR, such as misclassification and injection attacks on SSD models. This might be due to the inherent robustness of backdoor attacks in object detection models, as these models involve

Table 6. Mitigating backdoors in NLP.

Poisoned Model	Original		NAD		Ours	
	Acc.	ASR	Acc.	ASR	Acc.	ASR
Model-1	85%	92%	85%	29%	84%	18%
Model-2	89%	94%	88%	27%	88%	17%
Model-3	87%	91%	88%	29%	87%	14%

Table 7. Ablation study on different design choices.

Method	Accuracy	ASR
Original	91.52%	81.36%
MARTINI	90.31%	1.41%
w/o $\mathcal{L}_{content}$	90.13%	38.86%
w/o $\mathcal{L}_{SSIM}$	90.36%	35.93%
w/o $\mathcal{L}_{norm}$	90.27%	43.44%
w/o $\mathcal{L}_{smooth}$	90.13%	32.04%

complex predictions of bounding boxes and labels. Nevertheless, the results still demonstrate the generalizability of MARTINI to object detection.

NLP Sentiment Analysis. The idea of our technique can be extended to defend against natural language processing (NLP) backdoors by leveraging a sentence-to-sentence model. A specially designed transformation layer can transform abstract features of sentences to embed backdoor effects. Adversarially training on the generated backdoor samples can then eliminate these NLP backdoors. We apply MARTINI to NLP sentiment analysis. We use a pre-trained DistilBERT as the generator, insert our transformation layer before the decoding layer, and adversarially train the model. We leverage three TrojAI-round5 poisoned models (injected with a phrase trigger), and the results are reported in Table 6. We adapt NAD from image classification to this setting. From the table, we observe that MARTINI reduces ASR to 16% on average, with a 1% accuracy degradation. The baseline NAD can only reduce ASR to 28%. This result shows that MARTINI has good potential for defending NLP backdoors. We leave further exploration to future work.

G. Ablation Study

MARTINI features a few important design choices. In this section, we aim to study these choices individually to better understand their contributions to the performance. In particular, we study the effects of four loss terms used in backdoor generation. The ablation study is conducted on a ResNet20 model with CIFAR-10, and the results are presented in Table 7. Row 1 denotes the original backdoored model and row 2 the final result of our method. Rows 3-6 present the results of excluding each loss term individually during backdoor mitigation.

Observe that $\mathcal{L}_{content}$ can boost the performance by 37.45%. This is because it constrains the difference of feature representations between backdoor samples and normal inputs. Without it, the backdoor samples can be too different from normal inputs internally and the model cannot learn the correct features. $\mathcal{L}_{SSIM}$ and $\mathcal{L}_{smooth}$ have similar ASR reduction. The SSIM score directly constrains the quality of generated backdoor samples looking similar to original inputs. $\mathcal{L}_{smooth}$ further smooths the backdoor samples to improve the generated quality. $\mathcal{L}_{norm}$ on the normalization layer is also quite important as it makes sure the normalized inputs are not far from the original distribution.

Table 8. Impact of hyperparameters.

z	1	2	3	4
Accuracy	69.26%	67.99%	**68.61%**	67.66%
ASR	17.00%	8.33%	**5.89%**	10.44%
λ_0	0.0005	0.001	0.002	0.005
Accuracy	68.59%	**68.61%**	67.49%	68.24%
ASR	16.56%	**5.89%**	11.11%	13.67%
λ_1	50	100	150	200
Accuracy	68.96%	**68.61%**	68.42%	68.49%
ASR	15.11%	**5.89%**	12.33%	9.33%
λ_2	0.03	0.05	0.1	0.2
Accuracy	68.76%	**68.61%**	68.44%	68.34%
ASR	9.44%	**5.89%**	6.67%	14.89%

Impact of Hyperparameters. We study the impact of four hyperparameters used in Equation 2 and Equation 8. The study is conducted on a backdoored model by DFST on STL-10, which has 72.18% accuracy and 98.67% ASR. The results are shown in Table 8. Observe that the impacts are small. Most settings can achieve good ASR reduction. In comparison, the lowest ASR achieved by the baselines is 48.44%. The best λs are chosen based on that all the loss terms are at the same scale as discussed below Equation 8. That is, the weighted loss value for each term shall be similar.