Appendix

A. Type-II 2D-DCT Algorithm

The type-II 2D-DCT is given by a function $D : \mathbb{R}^{N1\times N2} \rightarrow \mathbb{R}^{N1\times N2}$ that maps an image data $\{g_{x,y}\}$ to its frequency representation $D = \{D_{k_x,k_y}\}$ with $D_{k_x,k_y} = w(k_x)w(k_y)\sum_{x=0}^{N_1-1}\sum_{y=0}^{N_2-1} g_{x,y} cos\left[\frac{\pi}{N_1}(x+\frac{1}{2})k_x\right] cos\left[\frac{\pi}{N_2}(y+\frac{1}{2})k_y\right]$, for $\forall k_x = 0, 1, ..., N_1 - 1$ and $\forall k_y = 0, 1, ..., N_2 - 1$, where $w(0) = \sqrt{\frac{1}{4N}}$ and $w(k) = \sqrt{\frac{1}{2N}}$ for $k > 0$.

B. Visual Examples of Different Triggers

We provide the pair-to-pair comparisons of samples patched with different triggers' visual effects in the image and frequency domain. Figure 7, 8 illustrate the comparison of the attack cases over the GTSRB and the TSRD dataset. We can see severe high-frequency artifacts similar to the CIFAR-10 dataset results presented in Section 3.2. We also provide the pair-to-pair extended comparison of both the image and frequency domain visual effects on the evaluated CIFAR-10 and PubFig dataset in Figure 9, 10. Those results over different datasets and different triggers are provided here to further support the existence of persistent high-frequency artifacts of previous backdoor attacks in Section 3.2.

C. Visual Examples of the Random Puturbation used in Developing the Detector

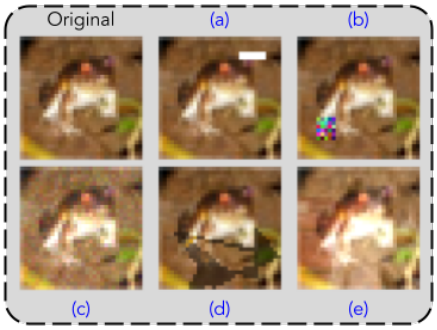

Figure 5: Visual examples of the random purturbations adopted in developing the detector. The upper left sample is a clean example, (a)-(e) are the perturbed results using different approaches.

Figure 5 presents the visual examples of the random perturbation results mentioned in Section 4.1. Figure 5 (a) is the example of patching a white rectangle of random size onto a random location of the image; Figure 5 (b) is the result of patching a rectangle of random size and random value to a random place. Those two random perturbations simulate patching localized triggers as mentioned and analyzed in Section 3.3. Figure 5 (c) is the visual result of adding random Gaussian noise; the result of drawing a random shadow of random shape is depicted in Figure 5 (d); finally, 5 (e) shows the visual result of random blend.

Note that the random perturbations used in Section 4.1 as illustrated here are of different shape and values from the tested triggers. We only use those random perturbations to simulate the resulting high-frequency artifacts using the two major patching methods, as analyzed in Section 3.3.

D. Linear Separability & Input Space

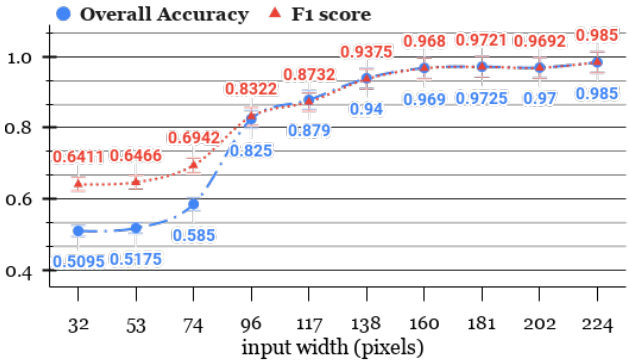

Figure 6: Detection Efficiency Using the Linear Model vs. Input Width

As mentioned in Section 4.1, we look into the relationship between the input space's size and linear models' efficiency. We test the F1-score and the linear models' overall accuracy on detecting triggered samples using different-input-spaced PubFig datasets. We test ten different values ranging from 32 to 224. The relationship between the input width and the detection efficiency is depicted in Figure 6. We can tell from the results that a larger-input-space samples can more easily be used to conduct a linear separation of the benign samples and the triggered samples. Meanwhile, the small-input-spaced samples are harder to be separated with linear models. Intuitively, we conduct the DCT of the whole image, thus acquiring a result of the same size as the image domain. So the larger input-spaced samples have more pixels representing the high-frequency coefficients, thus better reflecting the high-frequency artifacts when triggers are introduced. Based on the results shown in Figure 6 and as claimed in Section 4.1, an input space larger than 160 pixels can help linear models meet satisfying detection results.

E. DNN Model Architechures and Ablation Study

Given the different scales of difficulties to separate the DCT data in the frequency domain, we introduce a model ablation study to acquire the most simplistic DNN architecture that satisfies the detection performance to conduct the experiments in Section 4.1.

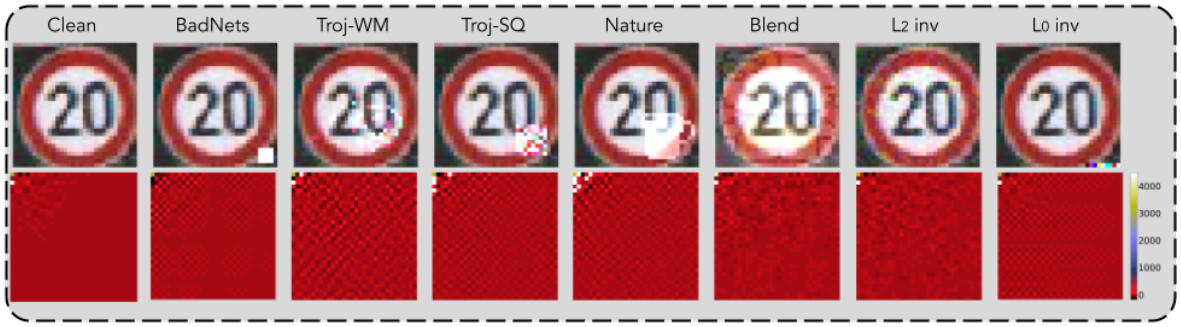

Figure 7: A pair-to-pair comparison of clean data and samples patching with different triggers on the GTSRB dataset. The frequency results are averaged over 10000 randomly selected samples from the test set.

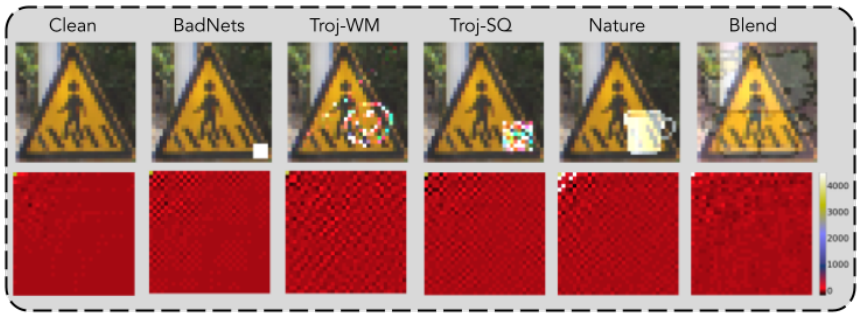

Figure 8: A pair-to-pair comparison of clean data and samples patching with different triggers on the TSRD database. The frequency results are averaged over all 4170 samples.

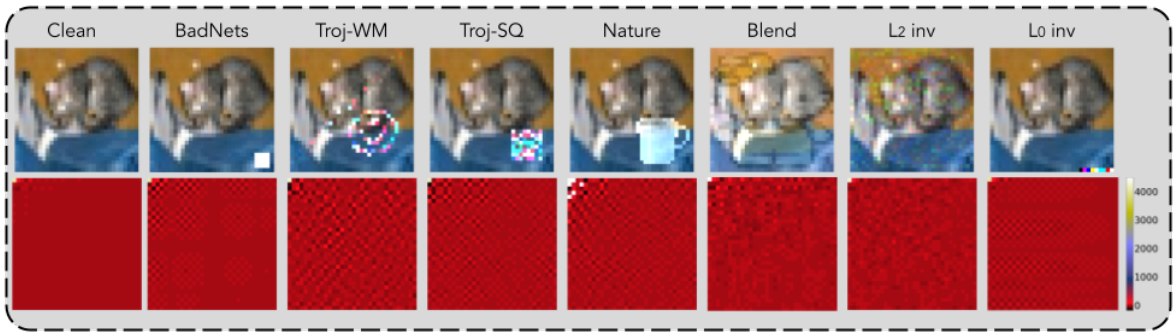

Figure 9: A pair-to-pair comparison of clean data and samples patching with different triggers on the Cifar10 dataset. The frequency results are averaged over 10000 randomly selected samples from the test set.

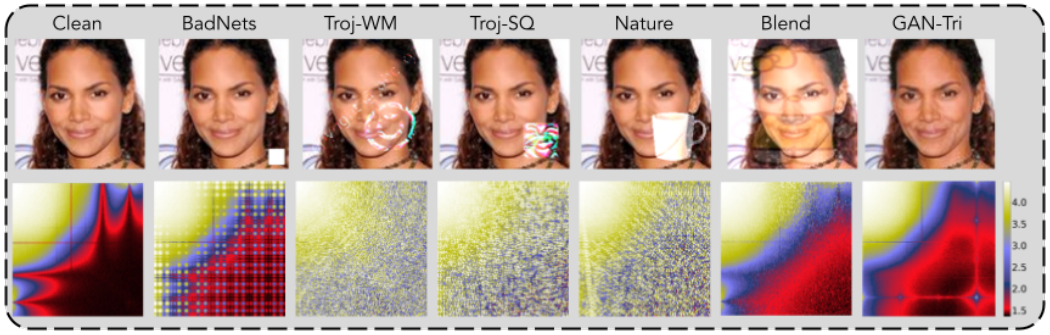

Figure 10: A pair-to-pair comparison of clean data and samples poisoned with different backdoor attacks on the PubFig dataset. The frequency results are averaged over 1000 randomly selected samples from the test set and clipped with the range of (1.5,4.5) for visualization.

Model	#Parameters	Train ACC	BadNets		Troj-WM		Troj-SQ		Nature		l_2 inv		l_0 inv	
			ACC	BDR	ACC	BDR	ACC	BDR	ACC	BDR	ACC	BDR	ACC	BDR
Linear	6,146	83.35	53.85	28.41	89.64	**100**	89.42	**99.56**	89.57	**99.85**	89.64	**100**	64.65	50.00
128-cell-hidden	393,602	88.23	54.80	21.89	**93.85**	**99.99**	**93.44**	**99.16**	**93.71**	**99.71**	**93.84**	**99.96**	55.61	23.50
3-layer CNN, $k_{max} = 32$	10,214	**95.55**	83.64	72.85	**97.21**	**99.99**	**96.94**	**99.47**	**97.09**	**99.76**	**97.21**	**99.99**	70.72	47.03
3-layer CNN, $k_{max} = 64$	31,862	**97.15**	84.26	71.72	**98.40**	**99.99**	**98.21**	**99.60**	**98.26**	**99.72**	**98.38**	**99.95**	55.71	14.61
3-layer CNN, $k_{max} = 128$	109,718	**98.36**	86.28	75.44	**98.55**	**99.99**	**98.40**	**99.68**	**98.40**	**99.67**	**98.55**	**99.99**	**97.46**	**97.80**
4-layer CNN, $k_{max} = 128$	245,014	**98.44**	87.63	78.18	**98.52**	**99.97**	**98.36**	**99.65**	**98.39**	**99.70**	**98.53**	**99.99**	**95.25**	**93.43**
5-layer CNN, $k_{max} = 128$	278,870	**98.58**	87.26	77.33	**98.52**	**99.97**	**98.38**	**99.57**	**98.44**	**99.69**	**98.58**	**99.96**	89.88	82.56
6-layer CNN, $k_{max} = 128$	292,002	**98.64**	**94.10**	**90.50**	**98.85**	**99.99**	**98.76**	**99.82**	**98.66**	**99.61**	**98.85**	**99.99**	**98.86**	**100**

Table 5: Model ablation study using the CIFAR-10 dataset. k_{max} represents the maximum value of the CNN kernels. We start the analysis from the most straightforward fully-connected linear model. Hidden layers, convolutional layers, or kernel sizes are gradually added or enlarged to test out the most simplistic model that can satisfy an outstanding detection efficiency. We present the training ACC, detection ACC, and BDR for each attack (%); the **boled** results are larger than 90%, which we interpret as satisfying results.

On large-input-spaced samples, namely the PubFig dataset, a linear model would already be able to achieve an outstanding detection efficiency which is introduced in Table 1, Section 4.2. Thus, no further ablation study is necessary for the large-input-space. The details of the linear model we adopted to conduct the detection task over the PubFig dataset are shown in Table 6. We use Adam with a learning rate of 0.01 as the optimizer for training this linear model. The binary cross-entropy is adopted as the loss function for the task of linear separation. We train the linear model with 50 epochs on the PubFig based dataset to attain the results shown in Table 1, Section 4.2.

Given that the DCT results in our evaluation have the same size as the original data's input space, the DCT results over small-input-space have a weaker ability to depict high-frequency artifacts compared to larger-input-space due to the limited number of high-frequency coefficients. Thus, as shown in Table 5, a similar fully connected linear model cannot meet a satisfying detection efficiency over the frequency domain using the same framework we proposed in this paper. We then conduct a thorough model ablation study by adding hidden layers or convolutional layers with different kernel sizes to obtain a most simplistic model that meets satisfying detection results over the evaluated attacks as shown in Table 5. With more complex architecture and parameters, the DNN can better detect the tested attacks. Based on the ablation study, we found that only until the model's architecture consists of 6 convolutional layers with $k_{max} = 128$ can it meet a satisfying and robust detection efficiency against all evaluated attacks.

Input $(224 \times 224 \times 3)$
Flatten (150528)
Dense (2)

Table 6: The network architecture of our simple Linear detector for large input space. We report the size of each layer.

The details of the simple 6-layer CNN detector for the small-input-space are explained in Table 7. The above experiments over the small-input-space are evaluated using this model to demonstrate the efficiency of conducting the detection of backdoor triggers in the frequency domain as elaborated in Section 4.2. We use Adam with a learning rate of 0.05 as the optimizer to train this model. Other settings are the same as the experiment conducted in large-input-space. The model took 150 epochs over the training set created using CIFAR-10 to converge and attain the results shown in Table 1, Section 4.2.

Input $(32 \times 32 \times 3)$
Conv2d 3×3 $(32 \times 32 \times 32)$
Conv2d 3×3 $(32 \times 32 \times 32)$
Max-Pooling 2×2 $(16 \times 16 \times 32)$
Conv2d 3×3 $(16 \times 16 \times 64)$
Conv2d 3×3 $(16 \times 16 \times 64)$
Max-Pooling 2×2 $(8 \times 8 \times 64)$
Conv2d 3×3 $(8 \times 8 \times 128)$
Conv2d 3×3 $(8 \times 8 \times 128)$
Max-Pooling 2×2 $(4 \times 4 \times 128)$
Flatten (2048)
Dense (2)

Table 7: The network architecture of our simple CNN detector for small-input-space. We report the size of each layer.

F. Target Model for Evaluating the Smooth Trigger

In Section 5.3, we evaluate the proposed smooth attack's attack efficiency on the CIFAR-10 and GTSRB dataset. As suggested in Algorithm 1, conducting the proposed attack requires a pre-trained model to generate the gradients for solving the bilevel optimization problem. We explain the details of the pre-trained model in Table 8. The model was trained using Adam optimizer with a learning rate at 0.05 for 150 epochs to converge. The base-line ACC over clean samples is 85.50% for the CIFAR-10 dataset. We also trained a base-line model on the GTSRB dataset for generating the smooth trigger over the GTSRB dataset. The GTSRB base-line model's ACC is 97.45%.

Input ($32 \times 32 \times 3$)
Conv2d 3×3 ($32 \times 32 \times 32$)
Conv2d 3×3 ($32 \times 32 \times 32$)
Max-Pooling 2×2 ($16 \times 16 \times 32$)
Conv2d 3×3 ($16 \times 16 \times 64$)
Conv2d 3×3 ($16 \times 16 \times 64$)
Max-Pooling 2×2 ($8 \times 8 \times 64$)
Conv2d 3×3 ($8 \times 8 \times 128$)
Conv2d 3×3 ($8 \times 8 \times 128$)
Max-Pooling 2×2 ($4 \times 4 \times 128$)
Flatten (2048)
Dense (10)

Table 8: The target model for evaluating the smooth trigger on Cifar10 and GTSRB dataset. We report the size of each layer.

G. Smooth Trigger on the GTSRB Dataset

As mentioned in 5.3, we conduct the smooth attack over the GTSRB dataset following the same pipeline as well. Figure 11 depicts the generated smooth trigger's visual results using the GTSRB dataset in the image and frequency domain. The dominant label computed using the Algorithm 1 is 1 on the GTSRB pre-trained model. Similar to the attack evaluation pipeline explained in Section 5.3, we conduct the backdoor attack with a poison rate of 0.1 over the target model using the GTSRB dataset. The model trained over the poisoned GTSRB dataset can maintain an ACC over clean samples at 97.42%, which is almost the same as the base-line model. Meanwhile, the ASR is 97.86% without defense. We observed the model could achieve an ASR greater than 90% even with one epoch of training. Meanwhile, the detection rate of the proposed detector in Section 4.1 can only achieve a BDR at 55.31% and an F1 score at 0.664 before considering this smooth attack. This detection efficiency can only drop the attack success rate of this GTSRB smooth trigger to 40.97%.

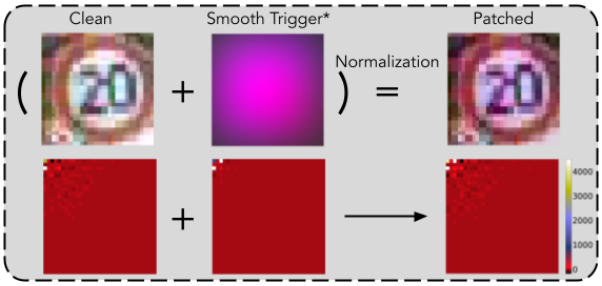

Figure 11: Visual effects over image and frequency domian of the smooth triggers. The trigger is multiplied by 5 for visualization. The right bottom depicts the heatmap averaged over 10000 samples patched with the smooth trigger. Both the trigger itself and the final images exhibit frequency spectra similar to natural images.

By incorporating this strongest smooth trigger found using Algorithm 1 into the development of the detector, we can regain a high efficient detection efficiency of a BDR at 85.53% and an F1 score of 0.8628 using the fine-tuning pipeline proposed in Section 5.4. This fine-tuning does not affect much over the other attack trigger's detection efficiency due to the limited scale as discussed in Section 5.4. Using this upgraded detector on the poisoned model, we can finally constrain the ASR from 97.86% to 13.27% by only adopting the detector to reject samples with triggers during the inference. In the case where we apply the detector to the training phase , we can further drop the ASR to 13.03%.

Overall, we observe very similar results to the attack conducted over the CIFAR-10 dataset. The GTSRB datasets' results further support the remarks mentioned in the paper and emphasize the importance of the frequency domain to the development of backdoor attacks and defenses.