Supplementary Material

Ren Wang[1], Gaoyuan Zhang[2], Sijia Liu[2], Pin-Yu Chen[2], Jinjun Xiong[2], and Meng Wang[1]

[1] Rensselaer Polytechnic Institute
[2] IBM Research
wangr8@rpi.edu, Gaoyuan.Zhang@ibm.com, Sijia.Liu@ibm.com, Pin-Yu.Chen@ibm.com, jinjun@us.ibm.com, wangm7@rpi.edu

1 Data-Limited TrojanNet Detector (DL-TND)

Visualization of neuron activation. DL-TND tests all the labels (classes) by calculating one universal perturbation and multiple per-image perturbations for each label. Each data sample can obtain a neuron activation vector with the universal perturbation and a neuron activation vector with its per-image perturbation. In Fig. S1, we show the neuron activation of five data samples with universal perturbations and per-image perturbations under a target label, a non-target label, and a label in a clean network (cleanNet). The output magnitude for each coordinate is represented using gray scale. One can see that the strong similarities only appear under the target label, which supports our motivation for the DL-TND.

Fig. S1. Output values of neuron activation with universal perturbation and per-image perturbation from five data samples. The first column shows the outputs corresponding to the target label. The second column shows the outputs corresponding to a non-target label in a Trojan network (TrojanNet). The third column shows the outputs corresponding to a label in a cleanNet. One can see that the strong similarities only appear under the target label, which supports the motivation for the data-limited Trojan detector.

Detection rule using median absolute deviation. Instead of using the detection rule in the main paper, we can also employ the median absolute deviation (MAD) method. By MAD, if a single value in the k-th position of $\frac{|(\mathbf{I})_{1/2}-\mathbf{I}|}{1.4826\cdot|(\mathbf{I})_{1/2}-\mathbf{I}|_{1/2}}$ is larger than 2 (provide 95% confidence rate), the network is poisoned and label k is a target label, where $\mathbf{I} = [I^{(1)}, I^{(2)}, \cdots, I^{(K)}]$. $|\cdot|$ represents the absolute value. $(\cdot)_{1/2}$ is the median of values in a vector. We compare DL-TND to Neural Cleanse (NC) [3] in Table S3 using MAD as the detection rule.

2 Data-Free TrojanNet Detector (DF-TND)

Visualization of logits output increase. Fig. S2 and S3 visualize the change of the logits output (defined in (8) in the main paper) of 10 data samples under a cleanNet and a TrojanNet when label 4 (lab4) is the target label. One can see that the minimum increase belonging to the target label is 600 while the maximum increase for labels in the cleanNet is 10. This large gap suggests that TrojanNets can be detected by properly selecting T_2 and there exists a wide selection range, implying the stability of our method.

Fig. S2. Visualization of logits output increase of 10 data samples using DF-TND on a TrojanNet when label 4 is the target label. The minimum increase belonging to the target label is 600

Fig. S3. Visualization of logits output increase of 10 data samples using DF-TND on a cleanNet. The maximum logits output increase is 10

3 A Unified Optimization Framework in TrojanNet Detection

In order to build TrojanNet detectors in both data-limited and data-free settings, we need to solve a sparsity-promoting optimization problem, in the specific forms of main paper's (2), (5), and (7), subject to a set of box and equality constraints.

In what follows, we propose a general optimization method by leveraging the idea of proximal gradient [1, 2].

Consider a problem with the generic form of problems (2), (5), and (7) in the main paper,

$$\min_{\mathbf{m},\boldsymbol{\delta},\mathbf{w}} F(\boldsymbol{\delta},\mathbf{m},\mathbf{w}) + \lambda\|\mathbf{m}\|_1 + \mathcal{I}(\boldsymbol{\delta}) + \mathcal{I}(\mathbf{m}) + \mathcal{I}'(\mathbf{w}), \tag{1}$$

where $F(\boldsymbol{\delta},\mathbf{m},\mathbf{w})$ denotes the smooth loss term, and $\mathcal{I}(\mathbf{x})$, $I'(\mathbf{w})$ denote the indicator functions to encode the hard constraints

$$\mathcal{I}(\mathbf{x}) = \begin{cases} 0 & \mathbf{x} \in [0,\alpha]^n \\ \infty & \text{otherwise,} \end{cases} \quad \mathcal{I}'(\mathbf{w}) = \begin{cases} 0 & \mathbf{w} \in [0,1]^n, \mathbf{1}^T\mathbf{w} = 1 \\ \infty & \text{otherwise.} \end{cases} \tag{2}$$

In $\mathcal{I}(\mathbf{x})$, $\alpha = 1$ for $\mathbf{m}$ and $\alpha = 255$ for $\boldsymbol{\delta}$. We remark that the binary constraint $\mathbf{m} \in \{0,1\}^n$ is relaxed to a continuous probabilistic box $\mathbf{m} \in [0,1]^n$.

To solve problem (1), we adopt the alternative proximal gradient algorithm [1], which splits the smooth-nonsmooth composite structure into a sequence of easier problems that can be solved more efficiently or even analytically. To be more specific, we alternatively perform

$$\mathbf{m}^{(t+1)} = \mathrm{Prox}_{\mu_t(\mathcal{I}+\lambda\|\cdot\|_1)}(\mathbf{m}^{(t)} - \mu_t\nabla_{\mathbf{m}}F(\boldsymbol{\delta}^{(t)},\mathbf{m}^{(t)})) \tag{3}$$

$$\boldsymbol{\delta}^{(t+1)} = \mathrm{Prox}_{\mu_t\mathcal{I}}(\boldsymbol{\delta}^{(t)} - \mu_t\nabla_{\boldsymbol{\delta}}F(\boldsymbol{\delta}^{(t)},\mathbf{m}^{(t+1)})) \tag{4}$$

$$\mathbf{w}^{(t+1)} = \mathrm{Prox}_{\mu_t\mathcal{I}'}(\mathbf{w}^{(t)} + \mu_t\nabla_{\mathbf{w}}F(\delta^{(t+1)},\mathbf{m}^{(t+1)},\mathbf{w}^{(t)})), \tag{5}$$

where μ_t denotes the learning rate at iteration t, and $\mathrm{Prox}_{\mu g}(\mathbf{a})$ denotes the proximal operator of function g with respect to the parameter μ at an input $\mathbf{a}$.

We next elaborate on the proximal operators used in (3)-(5). The proximal operator $\mathrm{Prox}_{\mu_t(\mathcal{I}+\lambda\|\cdot\|_1)}(\mathbf{a})$ is given by the solution to the problem

$$\min_{\mathbf{m}} \mathcal{I}(\mathbf{m}) + \lambda\|\mathbf{m}\|_1 + \frac{1}{2\mu_t}\|\mathbf{m}-\mathbf{a}\|_2^2, \tag{6}$$

where $\mathbf{a} := \mathbf{m}^{(t)} - \mu_t\nabla_{\mathbf{m}}F(\boldsymbol{\delta}^{(t)},\mathbf{m}^{(t)})$. The solution to problem (6), namely, $\mathbf{m}^{(t+1)}$ is given by [1]

$$m_i^{(t+1)} = \mathrm{Clip}_{[0,1]}(\mathrm{sign}(a_i)\max\{|a_i| - \lambda\mu_t, 0\}), \forall i, \tag{7}$$

where $m_i^{(t+1)}$ dentoes the ith entry of $\mathbf{m}^{(t+1)}$, and $\mathrm{Clip}_{[0,1]}$ is a clip function that clip the variable to 1 if it is larger than 1 and to 0 if it is smaller than 0. Similarly, $\boldsymbol{\delta}^{(t+1)}$ in (4) is obtained by

$$\delta_i^{(t+1)} = \mathrm{Clip}_{[0,255]}(b_i), \tag{8}$$

where $\mathbf{b} := \boldsymbol{\delta}^{(t)} - \mu_t\nabla_{\boldsymbol{\delta}}F(\boldsymbol{\delta}^{(t)},\mathbf{m}^{(t+1)})$.

The proximal operator $\mathrm{Prox}_{\mu_t\mathcal{I}'}(\mathbf{c})$ in (5) is given by the solution to the problem

$$\min_{\mathbf{w}} \mathcal{I}'(\mathbf{w}) + \frac{1}{2\mu_t}\|\mathbf{w}-\mathbf{c}\|_2^2, \tag{9}$$

which is equivalent to

$$\min_{\mathbf{w}} \ \|\mathbf{w}-\mathbf{c}\|_2^2, \quad \text{s.t.} \ \ \mathbf{0} \leq \mathbf{w} \leq \mathbf{1}, \ \mathbf{1}^T\mathbf{w} = 1. \tag{10}$$

Here $\mathbf{c} := \mathbf{w}^{(t)} + \mu_t \nabla_{\mathbf{w}} F(\delta^{(t+1)}, \mathbf{m}^{(t+1)}, \mathbf{w}^{(t)})$. The solution to problem (10) is given by [2]

$$\mathbf{w}^{(t+1)} = [\mathbf{c} - \mu\mathbf{1}]_+ , \tag{11}$$

where $[a]_+$ denotes the operation of $\max\{0, a\}$, and μ is the root of the equation $\mathbf{1}^T [\mathbf{c} - \mu\mathbf{1}]_+ = \sum_i \max\{0, c_i - \mu\} = 1$.

Substituting (7), (8) and (11) into (3)-(5), we then obtain the complete algorithm, in which each step has a *closed-form*.

4 DL-TND: Additional Experiments

Models for testing. Table S1 shows the numbers of different models used for testing. Models have three different architectures and are applied to CIFAR-10, GTSRB, and R-ImgNet. We trained 85 TrojanNets and 85 cleanNets, respectively. In addition to the diversity of model architecture and dataset types, we also train TrojanNets with different triggers. Table S2 shows the smallest test accuracy and attack success rate for TrojanNets and cleanNets. TrojanNets can reach a similar test accuracy as cleanNets while still keeping the high attack success rate. This suggests that they are valid TrojanNets as defined in Sec. 2.1 in the main paper.

Table S1. Numbers of different models for detection: Model structures include ResNet50, VGG16, AlexNet. Datasets include CIFAR-10, GTSRB, and R-ImgNet.

	CIFAR-10	GTSRB	R-ImgNet
ResNet50 (TrojanNet)	20	12	5
ResNet50 (cleanNet)	20	12	5
VGG16 (TrojanNet)	10	9	5
VGG16 (clean)	10	9	5
AlexNet (TrojanNet)	10	9	5
AlexNet (cleanNet)	10	9	5
Total	80	60	30

Applying median absolute deviation method as the detection rule. Table S3 provides the comparisons between DL-TND and NC method [3] on Trojan and cleanNets using Median Absolute Deviation (MAD) as the detection rule. Even using the MAD method as the detection rule, we find that DL-TND greatly outperforms NC in detection tasks of both TrojanNets and cleanNets.

Table S2. The smallest test accuracy and attack success rate for TrojanNets and cleanNets. TrojanNets can reach a similar test accuracy as cleanNets while still keeping the high attack success rate. This suggests that they are valid TrojanNets as defined in Sec. 2.1 in the main paper.

	CIFAR-10	GTSRB	R-ImgNet
Test accuracy (Trojan)	90.51%	92.99%	86.7%
Attack success rate	99.65%	99.65%	98.6%
Test accuracy (clean)	92.64%	92.5%	87.8%

Table S3. Comparisons between DL-TND and NC [3] on TrojanNets and cleanNets using Median Absolute Deviation as the detection rule (measured by number of correct detection/model number).

		DL-TND (clean)	DL-TND TND (poisoned)	NC (clean)	NC (poisoned)
CIFAR-10	ResNet-50	16/20	17/20	11/20	13/20
	VGG16	8/10	8/10	5/10	6/10
	AlexNet	8/10	8/10	6/10	7/10
GTSRB	ResNet-50	9/12	12/12	10/12	6/12
	VGG16	7/9	9/9	6/9	7/9
	AlexNet	7/9	9/9	5/9	5/9
ImageNet	ResNet-50	4/5	4/5	4/5	1/5
	VGG16	4/5	3/5	3/5	2/5
	AlexNet	4/5	4/5	4/5	1/5
Total		**67**/85	**74**/85	54/85	48/85

Varying number of data samples in each class. We also vary the number of validation data points for CIFAR-10 models and see the detection performance when we choose the quantile to be 0.5 (median). The number of data points in each class is chosen as 1, 2, and 5 and the corresponding AUC values are $0.96, 0.98$ and 1, respectively. We can see that the data-limited TrojanNet detector is effective even when only one data point is available for each class.

ROC curve for target label detection. Let the true positive rate be the detection success rate of target labels and the false negative rate be the detection error rate of cleanNets. Table 2 in the main paper also shows the AUC values for target label detection, and Fig. S4 shows the ROC curves for target label detection using DL-TND. We set $I^{(k)}$ to quantile-0.25, median, quantile-0.75 of the similarity values and vary T_1. Under the three different quantile selections, AUC values are all above 0.98.

Visualization of the universal perturbations. In Fig. S5, we show the universal perturbation obtained through (2) of the main paper. Due to the presence of backdoor in TrojanNets, universal perturbations can reveal *common patterns* with the real triggers, and this property is reflected in Fig. S5. Since DL-TND tries to find the smallest universal perturbation, the recovered perturbation pattern could be much more sparse than the Trojan trigger when the Trojan trigger

Fig. S4. ROC curve for target label detection using data-limited TrojanNet detector over 85 TrojanNets and 85 cleanNets

is very complicated. This can be viewed in the perturbation pattern in the last two columns of Fig. S5.

5 DF-TND: Additional Experiments

The sensitivity to trigger locations and sizes. Fig. S6 and Fig. S7 provide the experimental results for the sensitivity to trigger locations and sizes. Fig. S6 shows that locations of perturbations vary when the locations of Trojan triggers vary. However, the recovered perturbations do not always have the same locations as the Trojan triggers. Patterns shifted and enlarged due to the convolution operations. Fig. S7 shows that DF-TND can recover the trigger pattern when the size of the Trojan trigger increases, and the area of the recovered perturbation increases when the size of the Trojan trigger increases.

Improvements using the refine method - maximizing the neuron activation corresponding to the Trojan-related coordinate. Note that once the recovered data is obtained from the optimization problem (7) in the main paper, one can find the coordinate related to Trojan feature by checking the largest neuron activation value (or the largest weight) among all the coordinates. Then maximizing the output of the Trojan-related coordinate separately could provide a better result. Fig. S8 shows the improvements using DF-TND together with our refine method - maximizing the neuron activation corresponding to the Trojan-related coordinate. The refine method can increase the logits output belonging to the target label, while decrease the logits outputs belonging to the non-target labels simultaneously.

ROC curves for TrojanNet detection with clean validation inputs and random noise inputs. Fig. S9 (a) and (b) show the ROC curves for TrojanNet detection with clean validation inputs and random noise inputs, respectively. The true positive rate is the detection success rate for TrojanNets and the false

negative rate is the detection error rate for cleanNets. In both cases, DF-TND can reach nearly perfect AUC values 0.99. $T_2 = 55-400$ could provide a detection success rate of more than 85% for TrojanNets and a detection success rate of over 90% for cleanNets.

Recovered perturbations under different λ.

In Fig. S10 and Fig. S11, we vary the sparsity penalty parameter λ and obtain perturbations under a TrojanNet and a cleanNet. One can find that the trigger pattern appears in the perturbations under the TrojanNet. The perturbations under cleanNet behave like random noises. Another discovery is that the perturbations become more and more sparse when λ increases.

This method also works for random noise inputs. Fig. S11 shows the original noise images, trigger, perturbations under poisoned model, and perturbations under cleanNet. The same behaviors as the clean inputs are observed.

Fig. S5. Visualization of the universal perturbations obtained by our proposed DL-TND. Here the TrojanNets are trained by 10% poisoned data and clean data, respectively. First row: Seed input images (from left to right: 2 randomly selected CIFAR-10 images, 2 randomly selected GTSRB images, 2 randomly selected ImageNet images). Second row: Recovered images under cleanNets. Third row: Perturbation patterns given by the difference between the recovered images in the second row and the original seed image. Fourth row: Trojan triggers used for TrojanNets. Fifth row: Recovered images under TrojanNets. Sixth row: Perturbation patterns given by the difference between the recovered images in the fifth row and the original seed images. Perturbation has common patterns with the real Trojan triggers. Since DL-TND tries to find the smallest universal perturbation, the recovered perturbation pattern could be much more sparse than the Trojan trigger when the Trojan trigger is very complicated. This can be viewed in the perturbation pattern in the last two columns.

Fig. S6. Visualization of perturbations when locations of Trojan triggers change. Here the TrojanNets are trained by 10% poisoned data and clean data, respectively. First row: Trojan triggers in different locations (the centers of the triggers are listed above). Second row: Clean seed image 1 (ship). Third row: Recovered perturbations 1 with input from the second row. Fourth row: Clean seed image 2 (deer). Fifth row: Recovered perturbations 2 with input from the fourth row. Sixth row: random noise seed image 3. Seventh row: Recovered perturbations 3 with input from the sixth row. Eighth row: random noise seed image 4. Ninth row: Recovered perturbations 4 with input from the eighth row. Locations of perturbations vary when the locations of Trojan triggers vary. However, the recovered perturbations do not always have the same locations as the Trojan triggers. Patterns shifted and enlarged due to the convolution operations.

Fig. S7. Visualization of perturbations when sizes of Trojan triggers change. Here the Trojan ResNet-50 models are trained by 10% poisoned data and clean data, respectively. We vary the trigger size from 3×3 to 9×9 and show the recovery in different columns. First row: Trojan triggers with different sizes (the sizes of the triggers are listed above). Second row: Seed images (clean CIFAR-10 images and random noise images). Third row: Recovered images with inputs from the second row. Fourth row: Perturbation patterns given by the difference between the recovered images in the third row and the seed images. One can see that the area of the recovered perturbation increases when the size of the Trojan trigger increases.

(a) (b)

Fig. S8. Improvements using DF-TND together with our refine method - maximizing the neuron activation corresponding to the Trojan-related coordinate: (a) Visualization of logits output increase of 10 random noise inputs before using the refine method when label 4 is the target label. The maximum increase belonging to the target label is 600, the maximum decreasing belonging to the non-target label is 600. (b) Visualization of logits output increase of the same 10 random noise inputs after using the refine method when label 4 is the target label. The maximum increase belonging to the target label is 2500, the maximum decreasing belonging to the non-target label is 1500. The refine method increases the logits output belonging to the target label, while decreases the logits outputs belonging to the non-target labels

(a) (b)

Fig. S9. ROC curves for TrojanNet detection with clean validation inputs and random noise inputs: (a) Clean validation inputs (AUC=0.99) (b) Random noise inputs (AUC=0.99)

Fig. S10. Visualization of perturbations with CIFAR-10 image inputs when the sparsity penalty parameter λ varies. First row: Images under a TrojanNet. Second row: Images under a cleanNet.

Fig. S11. Visualization of perturbations with random noise inputs when the sparsity penalty parameter λ varies. First row: Images under a TrojanNet. Second row: Images under a cleanNet.

References

1. Bolte, J., Sabach, S., Teboulle, M.: Proximal alternating linearized minimization for nonconvex and nonsmooth problems. Mathematical Programming **146**(1-2), 459–494 (2014)
2. Parikh, N., Boyd, S., et al.: Proximal algorithms. Foundations and Trends® in Optimization **1**(3), 127–239 (2014)
3. Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., Zhao, B.Y.: Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks p. 0 (2019)