CHAPTER 15

Adversary detection

Another type of defense methods against evasion attacks, which we call *adversary detection*, aims to detect the existence of adversarial examples, rather than trying to classify them into the correct classes. The main assumption behind these methods is that the adversarial examples came from a different distribution compared with natural samples, so they could be detected with carefully designed detectors. In addition to evasion attacks, similar methodology can be applied to detect other types of adversarial attacks. In this section, we first discuss adversarial detection to testing time adversarial attacks. We then discuss detection methods against other attack scenarios.

15.1 Detecting adversarial inputs

A straightforward way toward adversarial example detection is to build a simple binary classifier separating the adversarial examples apart from the clean data. However, these methods usually suffer from lack of generalization to unforeseen attacks. Therefore an effective detection method is usually based on some characteristics of adversarial attacks instead of purely classifying an existing attack. In the following, we will introduce several popular detection methods.

Density estimation. A common observation is that adversarial examples often lie outside the natural image manifold, and many detection methods are based on these characteristics. In particular, Feinman et al. (2017) proposed to use kernel density estimation to measure how far a sample is from the provided data manifold. Letting $X = \{x_1, \ldots, x_n\}$ be training samples stored in the database, we can assume that the training data distribution is a mixture of Gaussians centered at those points, denoted as $p_X(x)$. Therefore, for a testing instance x, the density can be estimated as

$$p_X(x) = \frac{1}{n} \sum_{i=1}^{n} K_\sigma(x_i, x),$$

where $K_\sigma(\cdot, \cdot)$ is a kernel function. A commonly used kernel function is the Gaussian kernel, where $K_\sigma(x_i, x) = e^{-\frac{\|x_i - x\|^2}{2\sigma^2}}$. The KD-detection method proposed in (Feinman et al., 2017) fits the density function on each label. If

https://doi.org/10.1016/B978-0-12-824020-5.00026-0

an incoming example x is predicted as label y, we use the density estimation based on the samples from class y. Once $p_X(x)$ is computed, we can then get the statistics of KD estimation of natural examples versus adversarial examples to determine the threshold for detecting adversarial examples.

Many other algorithms also fall into this kernel density estimation framework. For example, Ma et al. (2018) observed that the local intrinsic dimension (LID) of hidden layer outputs differ between adversarial examples and natural examples, so they conduct density estimations on hidden layer features (instead of input layers) to detect adversarial examples. It is also observed in many works that an ensemble of several different layers can significantly boost the performance of adversarial detection in computer vision tasks, so most of the existing detectors are based on not only one, but on several hidden layer features. In another work, Lee et al. (2018) generated the class of conditional Gaussian distributions with respect to hidden layer output of the DNN under Gaussian discriminant analysis, which results in a confidence score based on the Mahalanobis distance (MAHA), followed by a logistic regression model on the confidence scores to detect adversarial examples. A joint statistical test pooling information from multiple layers is proposed by Raghuram et al. (2020) to detect adversarial examples.

Feature attribution methods for adversarial detection. In addition to density estimation based on hidden layer features, many works identify some other important features that can distinguish adversarial examples from natural examples. Here we will discuss an interesting finding that adversarial examples can be detected by feature attribution methods (Yang et al., 2020d).

Assume that the model is a function $f : \mathbb{R}^d \to [0, 1]^K$ and a feature attribution method ϕ maps an input image $x \in \mathbb{R}^d$ to a d-dimensional vector $\phi(x) \in \mathbb{R}^d$, where each element in $\phi(x)$ corresponds to how important the feature contributes to the model prediction. There are many feature attribution methods developed in the literature. For instance, the widely used leave-one-out (LOO) feature attribution method (Zeiler and Fergus, 2014; Li et al., 2016) measures the importance of each feature by the change of model output when removing the feature (e.g., setting it as 0). Mathematically, the LOO method designs ϕ as

$$\phi(x)_i = f(x)_c - f(x_{(i)})_c,$$

where c is the target class we are interested in, and $x_{(i)}$ means x with feature i removed.

Interestingly, Yang et al. (2020d) observed that adversarial examples have very different feature attribution maps from natural examples. As shown in Fig. 15.1, we observe that the feature attribution map for adversarial examples is much more spread than for natural examples, where for each plot we have the original example (top left figure) and the adversarial example generated by PGD attack (bottom left figure), and generate feature attribution map (e.g., using leave-one-out) of each figure. The pixel value distribution of each feature attribution map is then compared in the histogram on the bottom panel, showing very different patterns between original example and adversarial example. One hypothesis is that classifiers are very confident about the prediction based on certain feature when facing natural examples, but in adversarial examples, since we add noise to the input, the feature attributions become more spread since there is no particular pixel in the image leading to this false prediction. Similar behavior is also observed when applying other feature attribution maps, such as integrated gradient (IG) or other gradient-based feature attribution methods.

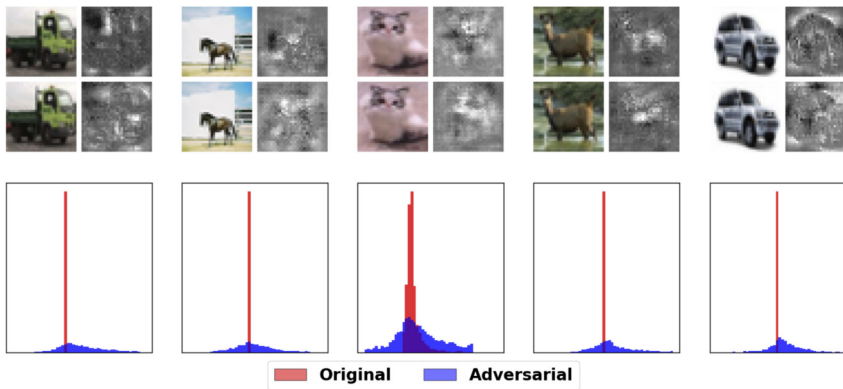

Figure 15.1 Feature attribution analysis in (Yang et al., 2020d). The first row shows the original CIFAR-10 examples and their corresponding feature attributions. The second row shows the adversarial examples and their corresponding feature attributions. The third row plots the histograms of the original and adversarial feature attributions. We can easily observe that adversarial examples have very different feature attribution maps from natural examples.

Motivated by this observation, we can measure the statistical dispersion in the feature attribution map to detect adversarial examples. In particular, Yang et al. (2020d) adopted several statistical measurements, including standard deviation, median absolute deviation (the median of absolute differences between entries and their median), and interquartile range (the difference between the 75th and 25th percentiles among all entries of fea-

ture attribution values). These values can serve as a good detection score for adversarial versus natural examples, as shown in Fig. 15.2. Using these scores and ensemble feature attribution maps on multiple layers, the ML-LOO detection algorithm proposed by Yang et al. (2020d) achieved very good performance on adversarial detection, compared with purely density-based methods.

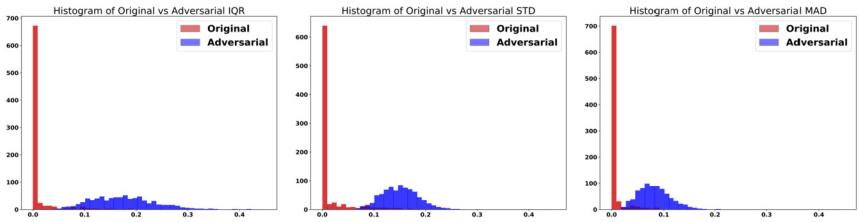

Figure 15.2 The dispersion scores (standard deviation, median absolute deviation, and interquartile range) for adversarial examples versus natural examples presented in (Yang et al., 2020d).

15.2 Detecting adversarial audio inputs

In general, detecting adversarial examples can be a challenging task. Many detection methods have been shown to be weakened or bypassed by advanced adversarial attacks that are aware of the defenses in place (Carlini and Wagner, 2017a).

However, in some data domains, we can leverage specific domain knowledge and data characteristics to detect adversarial inputs. In particular, for audio data inputs, Yang et al. (2019c) reveal the importance of using the temporal dependency in audio data to gain discriminate power against adversarial examples. Testing the automatic speech recognition (ASR) tasks and three recent audio adversarial attacks, they find that (i) input transformation developed from image adversarial defense provides limited robustness improvement and is subtle to advanced attacks and (ii) temporal dependency can be exploited to gain discriminative power against audio adversarial examples and is resistant to the considered adaptive attacks.

For (i), Yang et al. (2019c) show that four implemented transformation techniques on audio inputs, including waveform quantization, temporal smoothing, down-sampling, and autoencoder reformation, provide limited robustness improvement against the adaptive white-box attack proposed by Athalye et al. (2018), which aims to circumvent the gradient obfuscation issue incurred by input transformations.

For (ii), Yang et al. (2019c) propose a temporal dependency (TD) based detection method. Due to the fact that audio sequence has explicit temporal dependency (e.g., correlations in consecutive waveform segments), they explore if such temporal dependency will be affected by adversarial perturbations.

Methodology. The pipeline of the temporal dependency-based method is shown in Fig. 15.3. Given an audio sequence, they propose to select the first k portions of it (i.e., the prefix of length k) as input for ASR to obtain transcribed results as S_k. Moreover, they will also insert the whole sequence into ASR and select the prefix of length k of the transcribed result as $S_{\{\text{whole},k\}}$, which has the same length as S_k. We will then compare the consistency between S_k and $S_{\{\text{whole},k\}}$ in terms of temporal dependency distance. Here the word error rate (WER) is adopted as the distance metric (Levenshtein, 1966). For normal/benign audio instance, S_k and $S_{\{\text{whole},k\}}$ should be similar since the ASR model is consistent for different sections of a given sequence due to its temporal dependency. However, for audio adversarial examples, since the added perturbation aims to alter the ASR output toward the targeted transcription, it may fail to preserve the temporal information of the original sequence. Therefore, due to the loss of temporal dependency, S_k and $S_{\{\text{whole},k\}}$ in this case will not be able to produce consistent results. Based on such a hypothesis, they leverage the prefix of length k of the transcribed results and the transcribed k portion to potentially recognize adversarial inputs.

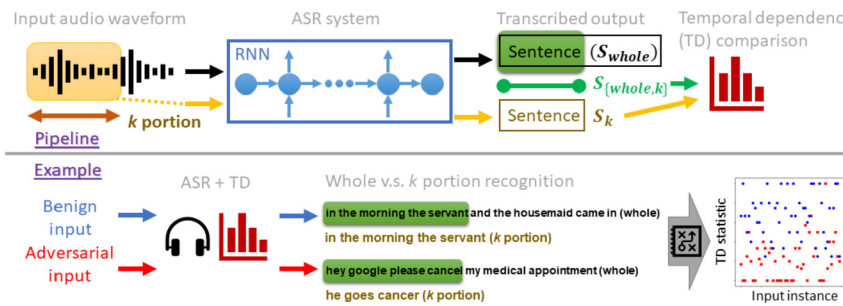

Figure 15.3 Pipeline and example of the proposed temporal dependency (TD) based method for discriminating audio adversarial examples in (Yang et al., 2019c).

Numerical experiments. Two attacks are considered for evaluation on the speech-to-text task.

Commander Song attack against speech-to-text translation (Commander). Commander Song (Yuan et al., 2018) is a speech-to-text targeted attack, which

can attack an audio extracted from a popular song. The adversarial audio can even be played over the air with its adversarial characteristics. The Kaldi speech recognition platform is used for analysis.

Optimization based attack against speech-to-text translation (Opt). The targeted speech-to-text attack proposed by Carlini and Wagner (2018) uses connectionist temporal classification (CTC) loss in a speech recognition system as an objective function and solves the task of adversarial attack as an optimization problem. The DeepSpeech speech-to-text transcription network is used for analysis, which is a biRNN-based model with beam search to decode text.

For detection methods, the standard evaluation metric is the area under curve (AUC) score, aiming to evaluate the detection efficiency. The proposed TD method focuses on how many adversarial instances are captured (true positive) without affecting benign instances (false positive). Therefore we follow the standard criteria and report AUC for TD. For the proposed TD method, we compare the temporal dependency based on WER, character error rate (CER), and the longest common prefix (LCP). LCP is a commonly used metric to evaluate the similarity between two strings. Given strings b_1 and b_2, the corresponding LCP is defined as $\max_{b_1[:k]=b_2[:k]} k$, where $[:k]$ represents the prefix of length k of a translated sentence.

In Commander Song attack, Yang et al. (2019c) directly examine whether the generated adversarial audio is consistent with its prefix of length k or not. Using TD method with $k = \frac{1}{2}$, all the generated adversarial samples showed inconsistency and thus were successfully detected. In Opt attack, as a baseline, Yang et al. (2019c) also directly train a one-layer LSTM with 64 hidden feature dimensions based on the collected adversarial and benign audio instances for classification. Some examples of translated results for benign and adversarial audios are shown in Table 15.1. They consider three types of adversarial targets: short – *hey google*; medium – *this is an adversarial example*; and long – *hey google please cancel my medical appointment*. The AUC score for these detection results for $k = 1/2$ in Table 15.2.

We can see that by using WER as the detection metric the temporal dependency-based method can achieve AUC as high as 0.936 on Common Voice and 0.93 on LIBRIS. Yang et al. (2019c) also explored different values of k and observed that the results do not vary too much. When $k = 4/5$, the AUC score based on CER can reach 0.969, which shows that such temporal dependency-based method is indeed promising in terms of distinguishing adversarial instances. Notably, these results suggest that the

Table 15.1 Examples of the temporal dependency-based detection method.

Type	Transcribed results
Original	then good bye said the rats and they went home
the first half of Original	then good bye said the raps
Adversarial (short)	hey google
First half of Adversarial	he is
Adversarial (medium)	this is an adversarial example
First half of Adversarial	these on adequate
Adversarial (long)	hey google please cancel my medical appointment
First half of Adversarial	he goes cancer

Table 15.2 AUC results of the proposed temporal dependency method.

Dataset	LSTM	TD (WER)	TD (CER)	TD (LCP ratio)
Common Voice	0.712	**0.936**	0.916	0.859
LIBRIS	0.645	0.930	**0.933**	0.806

temporal dependency-based method would suggest an easy-to-implement but effective method for characterizing adversarial audio attacks.

15.3 Detecting Trojan models

Wang et al. (2020d) study the problem of the Trojan network (TrojanNet) detection in the data-scarce regime, where only the weights of a trained DNN are accessed by the detector. We refer to Chapter 5 for details regarding backdoor attacks. Wang et al. (2020d) propose a data-limited TrojanNet detector (TND), when only a few data samples are available for TrojanNet detection. They show that an effective data-limited TND can be established by exploring connections between Trojan attack and prediction-evasion adversarial attacks including per-sample attack and all-sample universal attack. In addition, they propose a data-free TND, which can detect a TrojanNet without accessing any data samples for convolutional neural networks. Wang et al. (2020d) show that such a TND can be built by leveraging the internal response of hidden neurons, which exhibits the Trojan behavior even at random noise inputs. Their work offers a practical tool for TrojanNet detection and addresses the challenge of *How to detect a TrojanNet when having access to training/testing data samples is restricted or not allowed?* This is a practical scenario because it is a common practice for machine learning to leverage a pretrained but potentially untrusted neural networks in downstream applications such as transfer learning or finetuing.

Take image classifiers as an example. For backdoored models, since arbitrary images can be misclassified as the same target label by TrojanNet when these inputs consisting of the Trojan trigger are used in data poisoning, Wang et al. (2020d) hypothesize that there exists a *shortcut* in TrojanNet, leading to *input-agnostic* misclassification. Their approach is motivated by exploiting the existing *shortcut* for the detection of TrojanNets.

Trojan perturbation. Given a neural network model, let $f \in \mathbb{R}^K$ be the mapping from the input space to the logits of K classes. Let f_y denote the logits value corresponding to class y. The final prediction is then given by $\text{argmax}_y f_y$. Let $r \in \mathbb{R}^d$ be the mapping from the input space to neuron's representation defined by the output of the penultimate layer (namely, prior to the fully connected block of the model). Given a clean data $\mathbf{x} \in \mathbb{R}^n$, the poisoned data through *Trojan perturbation* $\boldsymbol{\delta}$ is then formulated as

$$\hat{\mathbf{x}}(\mathbf{m}, \boldsymbol{\delta}) = (1 - \mathbf{m}) \cdot \mathbf{x} + \mathbf{m} \cdot \boldsymbol{\delta}, \tag{15.1}$$

where $\boldsymbol{\delta} \in \mathbb{R}^n$ denotes pixelwise perturbations, $\mathbf{m} \in \{0, 1\}^n$ is a binary mask to encode the position where a Trojan stamp is placed, and $\cdot$ denotes the elementwise product. In trigger-driven Trojan attacks (Gu et al., 2017; Chen et al., 2017b; Yao et al., 2019), the poisoned training data $\hat{\mathbf{x}}(\mathbf{m}, \boldsymbol{\delta})$ is mislabeled to a target class to enforce a backdoor during model training. In clean-label Trojan attacks (Shafahi et al., 2018; Zhu et al., 2019a), the variables $(\mathbf{m}, \boldsymbol{\delta})$ are designed to misalign the feature representation $r(\hat{\mathbf{x}}(\mathbf{m}, \boldsymbol{\delta}))$ with $r(\mathbf{x})$ but without perturbing the label of the poisoned training data. We call this model a TrojanNet if it is trained over poisoned training data given by (15.1).

Data-limited TrojanNet detector (DL-TND): Wang et al. (2020d) address the problem of TrojanNet detection with the prior knowledge on model weights and a few clean test images, at least one sample per class. Let $\mathcal{D}_k$ denote the set of data within the (predicted) class k, and let $\mathcal{D}_{k-}$ denote the set of data with prediction labels different from k. Wang et al. (2020d) propose to design a detector by exploring how the per-image adversarial perturbation is coupled with the universal perturbation due to the presence of backdoor in TrojanNets. The rationale behind that is the per-image and universal perturbations would maintain a strong similarity while perturbing images toward the Trojan target class due to the existence of a Trojan shortcut. The framework is illustrated in Fig. 15.4(a).

Untargeted universal perturbation. Given images $\{\mathbf{x}_i \in \mathcal{D}_{k-}\}$, the goal is to find a *universal perturbation* tuple $\mathbf{u}^{(k)} = (\mathbf{m}^{(k)}, \boldsymbol{\delta}^{(k)})$ such that the predictions

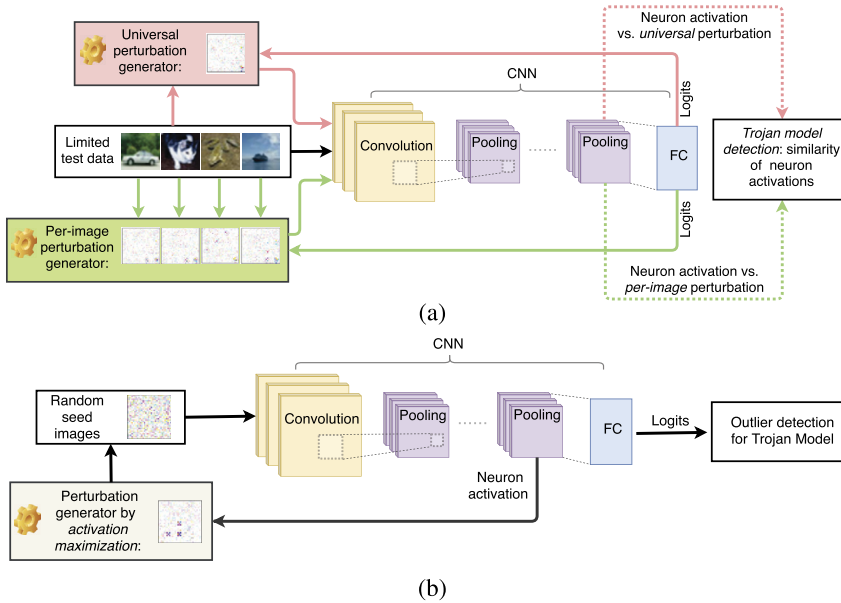

Figure 15.4 Frameworks of two detectors proposed in (Wang et al., 2020d): (a) data-limited TrojanNet detector (DL-TND); (b) data-free TrojanNet detector (DF-TND).

of these images in $\mathcal{D}_{k-}$ are *altered* given the current model. However, we require $\mathbf{u}^{(k)}$ not to alter the prediction of images belonging to class k, namely, $\{\mathbf{x}_i \in \mathcal{D}_k\}$. Spurred by that, the design of $\mathbf{u}^{(k)} = (\mathbf{m}^{(k)}, \boldsymbol{\delta}^{(k)})$ can be cast as the following optimization problem:

$$\begin{aligned} \underset{\mathbf{m},\boldsymbol{\delta}}{\text{minimize}} \quad & \ell_{\text{atk}}(\hat{\mathbf{x}}(\mathbf{m},\boldsymbol{\delta}); \mathcal{D}_{k-}) + \bar{\ell}_{\text{atk}}(\hat{\mathbf{x}}(\mathbf{m},\boldsymbol{\delta}); \mathcal{D}_k) + \lambda \|\mathbf{m}\|_1 \\ \text{subject to} \quad & \{\boldsymbol{\delta}, \mathbf{m}\} \in \mathcal{C}, \end{aligned} \tag{15.2}$$

where $\hat{\mathbf{x}}(\mathbf{m}, \boldsymbol{\delta})$ was defined in (15.1), $\lambda \geq 0$ is a regularization parameter that strikes a balance between the loss term $\ell_{\text{uatk}} + \bar{\ell}_{atk}$ and the sparsity of the trigger pattern $\|\mathbf{m}\|_1$, and $\mathcal{C} = \{\mathbf{0} \leq \boldsymbol{\delta} \leq 255, \mathbf{m} \in \{0,1\}^n\}$ is the constraint set of optimization variables $\mathbf{m}$ and $\boldsymbol{\delta}$, where [0,255] is the range of the RGB pixel values.

We next elaborate on the loss terms ℓ_{atk} and $\bar{\ell}_{\text{atk}}$ in problem (15.2). First, the loss ℓ_{atk} enforces to alter the prediction labels of images in $\mathcal{D}_{k-}$, and is defined as the C&W *untargeted* attack loss (Carlini and Wagner, 2017a)

$$\ell_{\text{atk}}(\hat{\mathbf{x}}(\mathbf{m},\boldsymbol{\delta}); \mathcal{D}_{k-}) = \sum_{\mathbf{x}_i \in \mathcal{D}_{k-}} \max\{f_{y_i}(\hat{\mathbf{x}}_i(\mathbf{m},\boldsymbol{\delta})) - \max_{t \neq y_i} f_t(\hat{\mathbf{x}}_i(\mathbf{m},\boldsymbol{\delta})), -\tau\}, \tag{15.3}$$

where y_i denotes the prediction label of $\mathbf{x}_i$ (recall that $f_t(\hat{\mathbf{x}}_i(\mathbf{m}, \boldsymbol{\delta}))$ denotes the logit value of the class t with respect to the input $\hat{\mathbf{x}}_i(\mathbf{m}, \boldsymbol{\delta})$), and $\tau \geq 0$ is a given constant that characterizes the attack confidence. The rationale behind $\max\{f_{y_i}(\hat{\mathbf{x}}_i(\mathbf{m}, \boldsymbol{\delta})) - \max_{t \neq y_i} f_t(\hat{\mathbf{x}}_i(\mathbf{m}, \boldsymbol{\delta})), -\tau\}$ is that it reaches a negative value (with minimum $-\tau$) if the perturbed input $\hat{\mathbf{x}}_i(\mathbf{m}, \boldsymbol{\delta})$ is able to change the original label y_i. Thus the minimization of ℓ_{atk} enforces the ensemble of successful label change of images in $\mathcal{D}_{k-}$. Second, the loss $\bar{\ell}_{\text{atk}}$ in (15.2) is proposed to enforce the universal perturbation *not* to change the prediction of images in $\mathcal{D}_k$. This yields

$$\bar{\ell}_{\text{atk}}(\hat{\mathbf{x}}(\mathbf{m}, \boldsymbol{\delta}); \mathcal{D}_k) = \sum_{\mathbf{x}_i \in \mathcal{D}_k} \max\{\max_{t \neq k} f_t(\hat{\mathbf{x}}_i(\mathbf{m}, \boldsymbol{\delta})) - f_{y_i}(\hat{\mathbf{x}}_i(\mathbf{m}, \boldsymbol{\delta})), -\tau\}, \quad (15.4)$$

where recall that $y_i = k$ for $\mathbf{x}_i \in \mathcal{D}_k$. We present the rationale behind (15.3) and (15.4) as follows. Suppose that k is a target label of Trojan attack. Then the presence of backdoor would enforce the perturbed images of non-k class in (15.3) toward being predicted as the target label k. However, the universal perturbation (performed like a Trojan trigger) would not affect images within the target class k, as characterized by (15.4).

Targeted per-image perturbation. If a label k is the target label specified by the Trojan adversary, then we hypothesize that perturbing each image in $\mathcal{D}_{k-}$ toward the target class k could go through the similar Trojan shortcut as the universal adversarial examples found in (15.2). Spurred by that, we generate the following targeted per-image adversarial perturbation for $\mathbf{x}_i \in \mathcal{D}_k$:

$$\underset{\mathbf{m}, \boldsymbol{\delta}}{\text{minimize}} \quad \ell'_{\text{atk}}(\hat{\mathbf{x}}(\mathbf{m}, \boldsymbol{\delta}); \mathbf{x}_i) + \lambda \|\mathbf{m}\|_1 \quad \text{subject to} \quad \{\boldsymbol{\delta}, \mathbf{m}\} \in \mathcal{C}, \quad (15.5)$$

where $\ell'_{\text{atk}}(\hat{\mathbf{x}}(\mathbf{m}, \boldsymbol{\delta}); \mathbf{x}_i)$ is the targeted C&W attack loss (Carlini and Wagner, 2017a)

$$\ell'_{\text{atk}}(\hat{\mathbf{x}}(\mathbf{m}, \boldsymbol{\delta}); \mathbf{x}_i) = \sum_{\mathbf{x}_i \in \mathcal{D}_{k-}} \max\{\max_{t \neq k} f_t(\hat{\mathbf{x}}_i(\mathbf{m}, \boldsymbol{\delta})) - f_k(\hat{\mathbf{x}}_i(\mathbf{m}, \boldsymbol{\delta})), -\tau\}. \quad (15.6)$$

For each pair of label k and data $\mathbf{x}_i$, we can obtain a per-image perturbation tuple $\mathbf{s}^{(k,i)} = (\mathbf{m}^{(k,i)}, \boldsymbol{\delta}^{(k,i)})$.

For solving both problems of universal perturbation generation (15.2) and per-image perturbation generation (15.5), the promotion of λ enforces a sparse perturbation mask $\mathbf{m}$. This is desired when the Trojan trigger is of small size. When the Trojan trigger might not be sparse, multiple values

of λ can also be used to generate different sets of adversarial perturbations. The proposed TrojanNet detector will then be conducted to examine every set of adversarial perturbations.

DL-TND detection rule. Let $\hat{\mathbf{x}}_i(\mathbf{u}^{(k)})$ and $\hat{\mathbf{x}}_i(\mathbf{s}^{(k,i)})$ denote the adversarial example of $\mathbf{x}_i$ under the universal perturbation $\mathbf{u}^{(k)}$ and the imagewise perturbation $\mathbf{s}^{(k,i)}$, respectively. If k is the target label of the Trojan attack, then based on our similarity hypothesis, $\mathbf{u}^{(k)}$ and $\mathbf{s}^{(k,i)}$ would share a strong similarity in fooling the decision of the CNN model due to the presence of backdoor. We evaluate such a similarity from the neuron representation against $\hat{\mathbf{x}}_i(\mathbf{u}^{(k)})$ and $\hat{\mathbf{x}}_i(\mathbf{s}^{(k,i)})$, given by $v_i^{(k)} = \cos\left(r(\hat{\mathbf{x}}_i(\mathbf{u}^{(k)})), r(\hat{\mathbf{x}}_i(\mathbf{s}^{(k,i)}))\right)$, where $\cos(\cdot,\cdot)$ represents cosine similarity. Here recall that r denotes the mapping from the input image to the neuron representation in CNN. For any $\mathbf{x}_i \in D_{k-}$, we form the vector of similarity scores $\mathbf{v}_{\text{sim}}^{(k)} = \{v_i^{(k)}\}_i$.

Given the similarity scores $\mathbf{v}_{\text{sim}}^{(k)}$ for each label k, we detect whether or not the model is a TrojanNet (and thus k is the target class) by calculating the so-called detection index $I^{(k)}$, given by the q%-percentile of $\mathbf{v}_{\text{sim}}^{(k)}$. The decision for TrojanNet is then made by $I^{(k)} \geq T_1$ for a given threshold T_1, and accordingly k is the target label. We can also employ the median absolute deviation (MAD) method to $\mathbf{v}_{\text{sim}}^{(k)}$ to mitigate the manual specification of T_1.

Data-free TrojanNet detector (DF-TND). The framework of data-free TrojanNet detector is summarized in Fig. 15.4(b). It was previously shown by Cheng et al. (2020a) and Wang et al. (2019b) that a TrojanNet exhibits an unexpectedly high neuron activation at certain coordinates. That is because the TrojanNet produces robust representation toward the input-agnostic misclassification induced by the backdoor. Given a clean data $\mathbf{x}$, let $r_i(\mathbf{x})$ denote the ith coordinate of the neuron activation vector. Motivated by (Engstrom et al., 2019a; Fong et al., 2019; Wang et al., 2020d), we study whether or not an inverted image that maximizes neuron activation is able to reveal the characteristics of the Trojan signature from model weights. We formulate the inverted image as $\hat{\mathbf{x}}(\mathbf{m}, \boldsymbol{\delta})$ in (15.1), parameterized by the pixel-level perturbations $\boldsymbol{\delta}$ and the binary mask $\mathbf{m}$ with respect to $\mathbf{x}$. To find $\hat{\mathbf{x}}(\mathbf{m}, \boldsymbol{\delta})$, we solve the problem of activation maximization

$$\begin{array}{ll} \underset{\mathbf{m},\boldsymbol{\delta},\mathbf{w}}{\text{maximize}} & \sum_{i=1}^{d} \left[w_i r_i(\hat{\mathbf{x}}(\mathbf{m}, \boldsymbol{\delta})) \right] - \lambda \|\mathbf{m}\|_1 \\ \text{subject to} & \{\boldsymbol{\delta}, \mathbf{m}\} \in \mathcal{C}, \mathbf{0} \leq \mathbf{w} \leq \mathbf{1}, \mathbf{1}^T \mathbf{w} = 1, \end{array} \tag{15.7}$$

where the notations follow (15.2) except the newly introduced variables $\mathbf{w}$, which adjust the importance of neuron coordinates. Note that if $\mathbf{w} = \mathbf{1}/d$,

then the first loss term in (15.7) becomes the average of coordinatewise neuron activation. However, since the Trojan-relevant coordinates are expected to make larger impacts, the corresponding variables w_i are desired for more penalization. In this sense the introduction of self-adjusted variables $\mathbf{w}$ helps us to avoid the manual selection of neuron coordinates that are most relevant to the backdoor.

DF-TND detection rule. Let the vector tuple $\mathbf{p}^{(i)} = (\mathbf{m}^{(i)}, \boldsymbol{\delta}^{(i)})$ be a solution of problem (15.7) given at a random input $\mathbf{x}_i$ for $i \in \{1, 2, \ldots, N\}$. Here N denotes the number of random images used in TrojanNet detection. We then detect if a model is TrojanNet by investigating the change of logits outputs with respect to $\mathbf{x}_i$ and $\hat{\mathbf{x}}_i(\mathbf{p}^{(i)})$, respectively. For each label $k \in [K]$, we obtain

$$L_k = \frac{1}{N} \sum_{i}^{N} [f_k(\hat{\mathbf{x}}_i(\mathbf{p}^{(i)})) - f_k(\mathbf{x}_i)]. \tag{15.8}$$

The decision of TrojanNet with the target label k is then made according to $L_k \geq T_2$ for a given threshold T_2.

Numerical experiments. As reported by Wang et al. (2020d), the neural network models include VGG16 (Simonyan and Zisserman, 2014), ResNet-50 (He et al., 2016), and AlexNet (Krizhevsky et al., 2012). Datasets include CIFAR-10 (Krizhevsky et al., 2009), GTSRB (Stallkamp et al., 2012), and Restricted ImageNet (R-ImgNet) (restricting ImageNet (Deng et al., 2009) to 9 classes). Wang et al. (2020d) trained 85 TrojanNets and 85 clean networks. The backdoor process includes different trigger patterns and poisoning data ratios. DL-TND is compared with the baseline Neural Cleanse (NC) (Wang et al., 2019b) for detecting TrojanNets.

To build DL-TND, Wang et al. (2020d) use 5 validation data points for each class of CIFAR-10 and R-ImgNet, and 2 validation data points for each class of GTSRB. They set $I^{(k)}$ to quantile-0.25, median, and quantile-0.75 and vary T_1. Let the true positive rate be the detection success rate for TrojanNets, and let the false negative rate be the detection error rate for cleanNets. Then the area under the curve (AUC) of receiver operating characteristics (ROC) can be used to measure the performance of the detection. Table 15.3 shows the AUC values, where "Total" refers to the collection of all models from different datasets. The results show that DL-TND can perform well across different datasets and model architectures. Moreover, fixing $I^{(k)}$ as the median, $T_1 = 0.54 \sim 0.896$ could provide a detection success rate over 76.5% for TrojanNets and a detection success rate over 82% for cleanNets.

Table 15.3 AUC values for TrojanNet detection and target label detection, given in the format $(\cdot, \cdot)$. The detection index for each class is selected as quantile Q = 0.25, Q = 0.5, and Q = 0.75 of the similarity scores.

	CIFAR-10	GTSRB	R-ImgNet	Total
Q = 0.25	(1, 1)	(0.99, 0.99)	(1, 1)	(1, 0.99)
Q = 0.5	(1, 0.99)	(1, 1)	(1, 1)	(1, 0.99)
Q = 0.75	(1, 0.98)	(1, 1)	(0.99, 0.97)	(0.99, 0.98)

Table 15.4 Comparisons between DL-TND (Wang et al., 2020d) and NC (Wang et al., 2019b) on TrojanNets and cleanNets using $T_1 = 0.7$. The results are reported in the format (number of correctly detected models)/(total number of models).

		DL-TND (clean)	DL-TND (Trojan)	NC (clean)	NC (Trojan)
CIFAR-10	ResNet-50	20/20	20/20	11/20	13/20
	VGG16	10/10	9/10	5/10	6/10
	AlexNet	10/10	10/10	6/10	7/10
GTSRB	ResNet-50	12/12	12/12	10/12	6/12
	VGG16	9/9	9/9	6/9	7/9
	AlexNet	9/9	8/9	5/9	5/9
ImageNet	ResNet-50	5/5	5/5	4/5	1/5
	VGG16	5/5	4/5	3/5	2/5
	AlexNet	4/5	5/5	4/5	1/5
Total		**84**/85	**82**/85	54/85	48/85

Table 15.4 shows the comparisons of DL-TND to Neural Cleanse (NC) (Wang et al., 2019b) on TrojanNets and cleanNets ($T_1 = 0.7$). The results for DF-TND can be found in (Wang et al., 2020d).

Finally, after detecting a trained model has backdoors, we can apply the model sanitization technique proposed by Zhao et al. (2020a), which uses limited clean dataset to mitigate the Trojan effects while maintaining high clean accuracy.

15.4 Extended reading

- Yang et al. (2020a) study how self-attention U-Net can enhance the characterization of adversarial audio examples.