ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-6327-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICASSP49357.2023.10095092

FORENSICS FOR ADVERSARIAL MACHINE LEARNING THROUGH ATTACK MAPPING IDENTIFICATION

Allen Yan, Jinsub Kim, and Raviv Raich

School of EECS, Oregon State University, Corvallis, OR 97331-5501

ABSTRACT

This paper considers the problem of performing post-attack forensic analysis for a test-time attack on a machine learning model. A test-time attack can be represented as a mapping that receives a benign test example as the input and outputs a falsified version of it. Given a set of attacked examples in the post-attack time, our objective is to identify the correct attack mapping among the collection of candidate attack strategies with diverse objectives and constraints. We present an attack mapping identification method that utilizes a pre-attack example recovery mechanism as a feature extraction method. Using numerical experiments, we demonstrate the effectiveness of the proposed approach in detecting the correct attack mapping among a number of different candidate attack strategies.

Index Terms— Adversarial machine learning, forensic analysis, test time attack

1. INTRODUCTION

Machine learning models are frequently designed to solve data analysis, computer vision, language processing, and decision-making problem. Lately, machine learning models have seen increased applications in security-sensitive domains, such as autonomous vehicle control and financial management. The low error tolerance nature of such critical tasks has sparked concern regarding the susceptibility of machine learning models to adversarial attacks, which manipulate the input data in complex ways to degrade the performance of machine learning models [1,2]. Many state-of-the-art machine learning models have thus far been proven vulnerable against adversarial attacks, which motivated the research on defending machine models against such attacks [3,4].

When considering problems involving adversarial attacks, defense is often not the only important aspect. In many high-stake applications, conducting forensic analysis on the adversarial attack to uncover properties and goals of the adversary is often crucial. For instance, several high-profile cyber attacks such as the Stuxnet attack and the 2015 attack on Ukraine power grid were followed by the post-attack forensic analysis [5,6], which revealed important details of the attack mechanisms. While attacks on machine learning models are considered viable in various domains, post-attack forensic analysis for such attacks has not been studied extensively in the literature. To fill this gap, we propose a forensic analysis approach that can be used to identify the correct attack mechanism based on a set of attacked examples.

Related Works Security of machine learning has been an active research topic for the past two decades. Various types of attacks on a machine learning system have been studied, including training data poisoning [7,8] and test-time attacks [9–11]. The focus of this paper is the test-time attack where an adversary is assumed to be capable of falsifying the input data of a machine learning model in the test time. Several countermeasures have been proposed in the literature to improve robustness of machine learning models against test-time attacks, including neural network distillation [12], adversarial training [13, 14], and use of a randomized decision rule [15–18]. Post-attack forensic techniques have been developed to address cyber security concerns in various domains (e.g., power grid security [19], Internet of Things [20, 21], computer network [22], and ransomware attacks [23]) where the techniques can be used to reveal the attack details and the system vulnerabilities exploited by the attack. However, to the best of our knowledge, post-attack forensic analysis for an attack on a machine learning system has not been studied in the literature.

Summary of Contributions Given a set of input examples falsified by a test-time attack, we consider the problem of identifying the true attack mapping, which is used by the adversary to map any given benign example to the corresponding attacked example. A main challenge in this forensic task is that we have access to only the attacked examples; the pre-attack examples are assumed to be unavailable[1]. Our main contributions are as follows. First, we formulate the attack mapping identification problem as a multiple hypothesis testing problem. Second, we develop an attack mapping identification approach by introducing and leveraging on a pre-attack example recovery method as a feature extraction tool. Third, we demonstrate the efficacy of the proposed approach in identifying the attack mappings of various test-time attacks by performing numerical experiments.

2. PROBLEM FORMULATION

2.1. Background: Test-time Adversarial Attack Forensics

The problem of *adversarial machine learning attack forensics* involves the identification of one of several adversarial machine learning attacks from a set of attacked examples affected by the attack.

To elaborate on this setting, we begin by introducing a cost-constrained test-time attack scenario, with the cost indicating the effort needed by an adversary to perform the attack. We consider an unattacked example $x \in \mathbb{R}^d$. A cost-constrained attack can be described as a mapping $a(\cdot)$ from the space of original examples to a space of attacked examples, without exceeding the allowed cost

This work was supported in part by the United States Department of the Navy, Naval Engineering Education Consortium (Award number: N00174-19-1-0009) and the National Science Foundation under Grant CCF-2224150.

[1]This setup is intended to reflect the practical challenge that the original input data might not be accessible at all in the presence of a test-time attack. For instance, if the input data entries consist of real time measurements of certain signals, and the adversary falsifies the measurements by modifying the signals directly, it is practically difficult to retrieve the original data.

threshold ϵ:

$$a : \mathbb{R}^d \to \mathbb{R}^d \quad \text{s.t.} \quad d(a(x), x) \leq \epsilon \quad \text{for all } x,$$
$$d(x, y) = \|x - y\|$$

Note that different norms can be considered for $d(\cdot\,, \cdot\,)$. In test-time adversarial attacks, the training data is assumed intact and the attack is applied to the test data prior to the application of the prediction rule. We assume that an attacker applies the same attack to each example in a stream of incoming examples.

2.2. Test-time Adversary

In this paper, we consider a *white box* attack on a neural network model. Specifically, the adversary is assumed to possess full knowledge of the target classifier model $g : \mathbb{R}^d \to \mathbb{R}^{|\mathcal{Y}|}$, which maps an example in the input space to the corresponding logit scores with respect to each label in the label set $\mathcal{Y}$. Given the target classifier model $g(\cdot)$, the attack mapping $a(\cdot)$ (i.e., how $a(x)$ is computed for a given example x) is typically defined as follows [3, 13]:

$$a(x) = \underset{u:\|u-x\|\leq\epsilon}{\arg\max}\ \ell(g(u), \hat{y}) \tag{1}$$

where the objective function $\ell(\cdot, \cdot)$ is determined according to the adversary's goal, ϵ represents the cost constraint on the attack design, and $\hat{y}$ denotes the decision of the target classifier on x. Note that the attack mapping $a(\cdot)$ is essentially determined by the cost threshold ϵ and the choice of $\ell(\cdot, \cdot)$. Regarding the choice of $\ell(\cdot, \cdot)$, a number of options have been discussed in the literature to represent a few potential objectives of an adversary [11]. Among them, we will mainly consider two categories: untargeted attacks and targeted attacks.

Untargeted attacks aim to shift the attacked examples as far away from the class distribution of the original examples as possible, without considering the class distribution of the destination. Madry *et. al.* [13] consider the robustification of classifiers against the untargeted attack $a(\cdot)$ defined by the following loss function:

$$\ell(g(u); y) = -\sum_{i=1}^{M} I(i = y) \log\left(\frac{e^{g_i(u)}}{\sum_{j=1}^{M} e^{g_j(u)}}\right). \tag{2}$$

Specifically, this attack function seeks to manipulate the data in order to maximize the cross-entropy loss of the classifier prediction $g(u)$ with respect to the original class decision y.

An alternative to the aforementioned attack type is the targeted attack, which seeks to move examples toward the distribution of a target class selected by the attacker. Carlini & Wagner [11] introduced in their work several instances of targeted attacks. One of which can be redefined using the following loss function:

$$\ell(g(u); t) = \sum_{i=1}^{M} I(i = t) \log\left(\frac{e^{g_i(u)}}{\sum_{j=1}^{M} e^{g_j(u)}}\right). \tag{3}$$

In this case, the attack function aims to minimize the cross-entropy loss of the classifier prediction $g(u)$ with respect to the selected target class t.

Note that in addition to objective function $l(\cdot\,, \cdot)$, different choices of target class t, cost ϵ, and optimization algorithm could all lead to different attack types. In this paper, our goal is to accurately distinguish among multiple attack types by observing a set of attacked examples, all affected by the same attack type. For our experiment, we will limit the defining components of attack types to be $l(\cdot, \cdot)$ (untargeted vs. targeted) and attack cost ϵ.

2.3. Mathematical Problem Statement

Let $\mathcal{X} \subset \mathbb{R}^d$ be the example feature space. Let

$$a^{(k)}(\cdot) : \mathcal{X} \to \mathbb{R}^d$$

be the k-th attack mapping for $k = 1, 2, \ldots, K$. Next, assume that a set of N *i.i.d.* examples $\mathcal{D}_N = \{x^1, \ldots, x^N\} \subset \mathcal{X}$ is drawn based on an unknown distribution P_X defined over $\mathcal{X}$. An attack index k is selected in $\{1, 2, \ldots, K\}$ and the attack $a^{(k)}$ is applied to $\mathcal{D}_N$ to obtain $\mathcal{A}_N = \{\bar{x}^1, \ldots, \bar{x}^N\}$ where $\bar{x}^i = a^{(k)}(x^i)$ for $i = 1, \ldots, N$. Our goal is to find a mapping that maps $\mathcal{A}_N$ back to k accurately given only (i) the attack mappings: $a^{(1)}, \ldots, a^{(K)}$ and (ii) a training set $\mathcal{D}' \subset \mathcal{X}$ sampled *i.i.d.* from P_X independent of the test set $\mathcal{D}_N$. Note that for this problem, it is assumed that all K attack mappings are known.

3. THE PROPOSED SOLUTION

To classify the attack type from the set of attacked examples $\{\bar{x}^1, \ldots, \bar{x}^N\}$, we consider the following classification rule:

$$\hat{k} = \arg\min_k \frac{1}{N} \sum_{i=1}^{N} \|\bar{x}^i - a^{(k)}(\hat{x}_k^i)\|^2$$

where

$$\hat{x}_k^i = f^{(k)}(\bar{x}^i)$$

is the estimated pre-attack example generated by inverse attack model $f^{(k)}(\cdot)$. The inverse attack model $f^{(k)}(\cdot)$ is the approximate inverse mapping of attack $a^{(k)}(\cdot)$, trained to satisfy $\bar{x} \approx a^{(k)}(f^{(k)}(\bar{x}))$ for $\bar{x} \in \{a^{(k)}(x) | x \in \mathcal{D}'\}$.

The intuition behind this approach is based on the speculation that the inverse attack mappings of adversarial examples are uniquely defined by the characteristics of the attack strategy. If an adversarial example $\bar{x}^i$ is "undone" by applying the inverse attack mapping $f^{(k)}(\cdot)$ of an incorrect attack strategy, it would likely produce an incoherent pre-attack example $\hat{x}_k^i$ that, when attacked again by $a^{(k)}(\cdot)$, maps to a result with significant deviation from the true adversarial example $\bar{x}^i$. Following this intuition, the true attack strategy of an adversarial example could be detected by evaluating the fitting error between $\bar{x}^i$ and $a^{(k)}(\hat{x}_k^i)$ across a range of attack strategy hypotheses k for $k \in \{1, \ldots, K\}$.

As mentioned earlier, this approach relies on estimating a pre-attack example. To that end, we propose an autoencoder-based approach for learning the inverse attack mapping.

3.1. Pre-attack Example Reconstruction

In order to obtain a means to estimate the pre-attack examples according to a given attack strategy hypothesis, we consider an autoencoder architecture for implementing $f^{(k)}(\cdot) = h(\cdot\,; \theta_k)$, where $h(\cdot\,; \theta) : \mathbb{R}^d \to \mathbb{R}^d$ denotes an autoencoder function parameterized by θ. Different from a regular autoencoder training, in which input and output examples are identical, we consider input-output pairs of the form $(a^{(k)}(x), x)$ for $x \in \mathcal{D}'$.

Specifically, to train the inverse attack autoencoders $\{f^{(1)}(\cdot), \ldots, f^{(K)}(\cdot)\}$, we use set $\mathcal{D}'$ to construct K training sets consisting of attacked/pre-attack example pairs for each attack. The kth training set is given by $\mathcal{D}'_k = \{(a^{(k)}(x), x)\}_{x \in \mathcal{D}'}$. For each $k \in \{1, 2, \ldots, K\}$, the k-th inverse attack autoencoder $f^{(k)}(\cdot)$ is given by

$$f^{(k)}(\cdot) = h(\cdot, \theta_k^*)$$

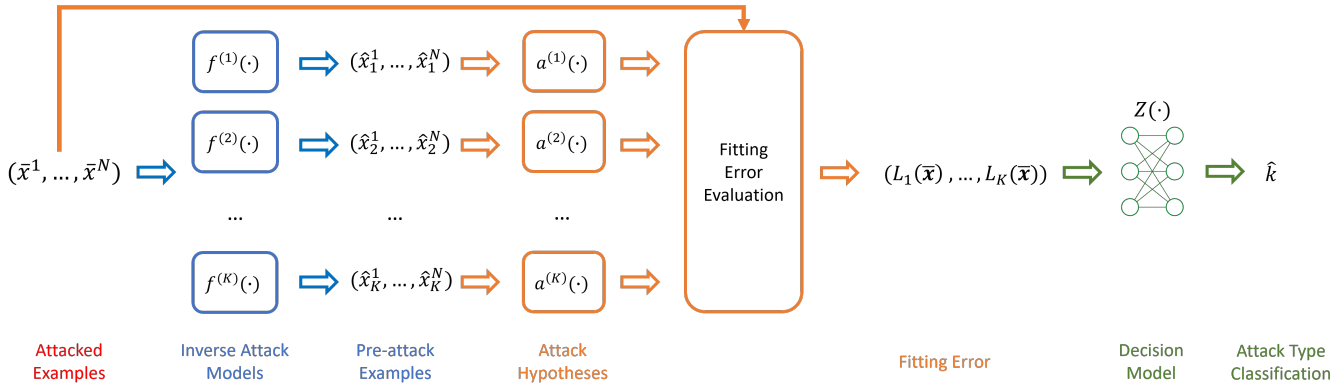

Fig. 1: Data flow of our proposed detection approach. Observed attacked examples are first "unattacked", then "re-attacked" using each attack mapping hypothesis to compute fitting errors, which are then passed to a decision model for classification.

where θ_k^* is obtained using dataset $\mathcal{D}'_k$ and the following training objective:

$$\theta_k^* = \arg\min_{\theta} \frac{1}{|\mathcal{D}'_k|} \sum_{(a,x)\in\mathcal{D}'_k} \|x - h(a;\theta)\|^2 .$$

3.2. Decision Model

In addition to the inverse attack models, a decision model is needed to aid the process of detection. While possession of the inverse attack models alone allows the fitting error

$$L_k(\bar{\mathbf{x}}) = \frac{1}{N}\sum_{i=1}^{N} \|\bar{x}^i - a^{(k)}(\hat{x}_k^i)\|^2 \tag{4}$$

to be computed, we noticed in our experiments that $L_k(\cdot)$ often shows significant shift in value range when the detection problem considers a large number of hypotheses, which prevents the direct use of $\arg\min$ function as an effective classification metric without a form of regularization. To resolve this, we chose to delegate the classification to a surrogate decision model $Z(\cdot)$ trained to make classification decision $\hat{k}$ using $\mathbf{L} = (L_1(\bar{\mathbf{x}}), \ldots, L_K(\bar{\mathbf{x}}))$ as inputs. For our problem, we chose a single-layer perceptron as the architecture for the decision model $Z(\cdot)$. The detailed steps for training $Z(\cdot)$, including generation of training data examples, are given in Algorithm 1. A complete diagram of the detection procedure is shown in Figure 1.

Algorithm 1: Training Algorithm of Decision Model

Input: $\{a^{(1)}(\cdot), \ldots, a^{(K)}(\cdot)\}$, $\{f^{(1)}(\cdot), \ldots, f^{(K)}(\cdot)\}$, $\mathcal{D}'$, N, T

for $i = 0, 1, \ldots, T/K$ **do**
- Select N examples without replacement from $\mathcal{D}'$: $\Rightarrow \{x^1, \ldots, x^N\}$;
- **for** $k = 1, 2, \ldots, K$ **do**
  - Apply $a^{(k)}(\cdot)$ to obtain $\{\bar{x}^1 = a^{(k)}(x^1), \ldots, \bar{x}^N = a^{(k)}(x^N)\}$;
  - $S \leftarrow S \cup (\{\bar{x}^1, \ldots, \bar{x}^N\}, k)$;

for $(\{\bar{x}^1, \ldots, \bar{x}^N\}, k) \in S$ **do**
- **for** $q = 1, 2, \ldots, K$ **do**
  - $L_q \leftarrow \frac{1}{N}\sum_{i=1}^{N} \|\bar{x}^i - a^{(q)}(f^{(q)}(\bar{x}^i))\|^2$;
- $\mathbf{L} = (L_1, \ldots, L_K)$;
- $\mathcal{L} \leftarrow \mathcal{L} \cup (\mathbf{L}, k)$;

Train the decision model $Z(\cdot)$ using training examples in $\mathcal{L}$.

4. NUMERICAL EXPERIMENTS

Experimental Setting: In this paper, we evaluated the performance of our solution approach on the MNIST handwritten digits dataset [24]. We select the range of evaluated attack types to be a combination of two attacker objectives (untargeted and targeted) and six cost constraint values $\epsilon \in \{1, \ldots, 6\}$ under the ℓ_2 distance metric proposed in Szegedy *et al.* [9], that is, $\|\bar{x}^i - x^i\| \leq \epsilon, \forall i \in \{1, \ldots, N\}$. This allows us a to form a total of $K = 12$ attack type hypotheses.

Attack implementation To implement untargeted attacks, we replicated the attacker objective used in the work of Madry *et al.* [13] shown in Equation (2). For targeted attacks, we used our redefinition of Carlini & Wagner's [11] attacker objective shown in Equation (3), and selected t to be the incorrect class with the highest prediction score for each example, that is,

$$t^i = \arg\max_{j\neq y} g_j(x^i).$$

To select a target model with reasonable degree of robustness for practicality, we replicated the robust optimization model for MNIST proposed in Madry *et al.* [13] (two convolutional layers, first with 32 filters, second with 64, both followed by a 2×2 max pooling layer, and a fully connected layer with size 1024 at the end) and trained it to defend against an ℓ_2 cost constraint of $\epsilon = 3$. Using the above settings, we implemented all selected attack strategies using projected gradient descent algorithm similar to the implementation presented in Madry *et al.* [13].

Inverse attack implementation Next, we designed an autoencoder architecture of our inverse attack models to consist of three convolutional layers, each with 8, 16, and 32 filters, respectively, followed by two linear layers with size of 128 and 50, respectively. Using this architecture, we trained an autoencoder $f^{(k)}(\cdot)$ following Section 3.1.

Decision model training To train our single-layer perceptron decision model, we followed the procedure described in Algorithm 1.

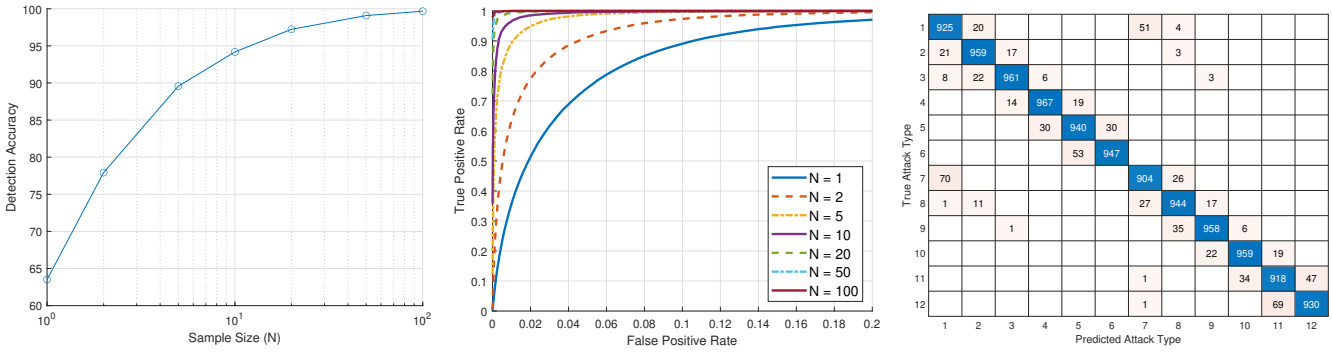

Fig. 2: Left: Detection accuracy as a function of sample size N. Middle: Macro-average ROC curve of the proposed decision model output for different values of the test sample sizes N. Right: Confusion matrix with sample size $N = 10$. Classes 1-6 correspond to untargeted attacks under ℓ_2 cost constraint of 1-6, and classes 7-12 correspond to targeted attacks in same order.

Performance evaluation During test time, the same data generation procedure described in Algorithm 1 is applied to the unattacked test dataset (separate from the training set $\mathcal{D}'$) to generate the fitting-error-attack-type pairs denoted by $\mathcal{L}$ in Algorithm 1. We use micro-averaged and macro-averaged ROC (AUC) as the evaluation metric for our approach, which are commonly used in multi-label learning problems [25]. To compute the averaged ROC (AUC), we generate test data following the training process describe in Algorithm 1 replacing the training data with withheld test data. Instead of training model $Z(\cdot)$, we apply the model to obtain test scores for each attack type. To obtain the ROC curve (or the AUC) for attack k, we consider a binary hypothesis testing in which data associated with k-attack is positive and all other data is considered negative and the kth test score of model Z is used a decision statistic. Finally the overall ROC curve (or AUC) is obtained by averaging the K ROC curves (AUC values).

Results & Analysis The following results are obtained by evaluating on 120000 instances of test data, which are obtained by applying the 12 candidate attack strategies to 10000 MNIST examples. Figure 2 (left) shows the accuracy of our detection approach as a function of the test sample size N. Our approach was able to achieve around 94% accuracy with sample size of 10.

To further assess the quality of the log-softmax output scores as a test statistics for the attack detection problem, we analyzed the averaged ROC curve as a function of sample size N. The ROC curves were obtained by averaging the one-vs-others ROC curve for each attack. The macro-average ROC for each sample size, N, is shown in Figure 2 (middle). We observe that, as expected, increasing the test sample size N drives the ROC curve towards to top-left corner, thereby increasing the area-under-the-ROC-curve (AUC) towards 1. The micro-averaged and macro-averaged ROC (AUC) computed from our results were nearly identical, which was expected since our experiment setting did not involve class imbalance. The corresponding averaged AUC values for each sample size N are provided in Table 1.

The confusion matrix shown in Figure 2 (right) shows that misdetections across attacker objectives (i.e., targeted vs. untargeted) are more prevalent for low cost attacks. But for high cost attacks, misdetections usually occur on the same attacker objective with a neighboring cost constraint. This suggests that different attacks become more distinct as the cost constraint increases, which we interpret as a beneficial trend because high-cost attacks are more detrimental and pose greater demand for correct identification of attacker objectives.

Overall, these results confirm that our proposed approach can effectively distinguish among multiple attack types, include ones that are similar in nature and only differ mildly in their attack cost. This also hints on the possibility of using the proposed approach for attack cost estimation.

Table 1: The micro (m) and macro-average (M) AUC for one-vs-rest detection of attack type as a function of the sample size N.

N	1	2	5	10
m. AUC	0.95876	0.98335	0.99565	0.99815
M. AUC	0.95831	0.98306	0.99552	0.99822
N	20	50	100	
m. AUC	0.99966	0.99989	0.99995	
M. AUC	0.99961	0.99989	0.99997	

5. CONCLUSION

In this paper, we presented the first study on post-attack forensics for test-time attacks on a machine learning model. We developed an attack mapping identification approach that can be used to detect the correct attack mapping, given the observation of attacked examples. Our numerical experiments demonstrated consistency of our detection approach. It is worth noting that our approach could detect the correct attack mapping with a high detection probability (around 0.94) by using only 10 attacked examples as the observation. This indicates that the proposed forensic approach can be potentially applied during the attack time to detect the correct mapping of the ongoing attack based on just a few attacked examples. In our future research, we plan to investigate how the knowledge of the attack mapping can be used to adapt the countermeasure in real time to improve the defense against the ongoing attack.

6. REFERENCES

[1] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z Morley Mao, "Adversarial sensor attack on lidar-based perception in autonomous driving," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019, pp. 2267–2281.

[2] Micah Goldblum, Avi Schwarzschild, Ankit Patel, and Tom Goldstein, "Adversarial attacks on machine learning systems

for high-frequency trading," in *Proceedings of the Second ACM International Conference on AI in Finance*, 2021, pp. 1–9.

[3] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin, "Black-box adversarial attacks with limited queries and information," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2137–2146.

[4] Moustapha M Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet, "Houdini: Fooling deep structured visual and speech recognition models with adversarial examples," *Advances in neural information processing systems*, vol. 30, 2017.

[5] Ralph Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, 2011.

[6] R. M. Lee, M. J. Assante, and T. Conway, "Analysis of the cyber attack on the ukrainian power grid," E-ISAC white paper, march 2016.

[7] Battista Biggio, Blaine Nelson, and Pavel Laskov, "Poisoning attacks against support vector machines," in *Proceedings of the 29th International Conference on Machine Learning*, 2012, p. 1467–1474.

[8] Shike Mei and Xiaojin Zhu, "Using machine teaching to identify optimal training-set attacks on machine learners," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2015, p. 2871–2877, AAAI Press.

[9] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations*, 2014.

[10] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[11] Nicholas Carlini and David Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*. Ieee, 2017, pp. 39–57.

[12] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *IEEE Symposium on Security and Privacy (SP)*, 2016, pp. 582–597.

[13] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018.

[14] Maksym Andriushchenko and Nicolas Flammarion, "Understanding and improving fast adversarial training," in *Advances in Neural Information Processing Systems*, 2020, vol. 33, pp. 16048–16059.

[15] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille, "Mitigating adversarial effects through randomization," in *International Conference on Learning Representations*, 2018.

[16] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten, "Countering adversarial images using input transformations," in *International Conference on Learning Representations*, 2018.

[17] Yuchen Zhang and Percy Liang, "Defending against whitebox adversarial attacks via randomized discretization," in *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*. 16–18 Apr 2019, pp. 684–693, PMLR.

[18] Xiao Wang, Siyue Wang, Pin-Yu Chen, Yanzhi Wang, Brian Kulis, Xue Lin, and Sang Peter Chin, "Protecting neural networks with hierarchical random switching: Towards better robustness-accuracy trade-off for stochastic defenses," in *International Joint Conference on Artificial Intelligence*, 2019, pp. 6013–6019.

[19] Melike Erol-Kantarci and Hussein T. Mouftah, "Smart grid forensic science: applications, challenges, and open issues," *IEEE Communications Magazine*, vol. 51, no. 1, pp. 68–74, 2013.

[20] Christopher Rondeau, Michael Temple, and C. S. Kabban, "Td-dna feature selection for discriminating wirelesshart iiot devices," in *Proceedings of the 53rd Hawaii International Conference on System Sciences*, 2020.

[21] T. J. Bihl, J. Schoenbeck, C. Rondeau, A. M. Jones, and Y. Adams, "Dna feature selection for discriminating wirelesshart iiot devices," in *Proceedings of the 54th Hawaii International Conference on System Sciences*, 2021.

[22] Christopher P. Lee and John A. Copeland, "Flowtag: A collaborative attack-analysis, reporting, and sharing tool for security researchers," in *Proceedings of the 3rd International Workshop on Visualization for Computer Security*, 2006, VizSEC '06, p. 103–108.

[23] Benjamin Reidys, Peng Liu, and Jian Huang, "Rssd: Defend against ransomware with hardware-isolated network-storage codesign and post-attack analysis," in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2022, ASPLOS '22, p. 726–739.

[24] Yann LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[25] Min-Ling Zhang and Zhi-Hua Zhou, "A review on multi-label learning algorithms," *IEEE transactions on knowledge and data engineering*, vol. 26, no. 8, pp. 1819–1837, 2013.