Designing Adversarial Attacks against Machine Learning Systems: an Optimization Framework

Jirong Yi, Zain Khan, Raghuraman Mudumbai, Weiyu Xu
Department of Electrical and Computer Engineering, University of Iowa, Iowa City, IA, USA, 52242
Email: {jirong-yi, zain-khan, weiyu-xu}@uiowa.edu, rmudumbai@engineering.uiowa.edu

Abstract—**In this paper, we consider the problem of attacking a machine learning system (MLS) without knowing its system parameters. We model the problem of designing the most effective attack under uncertainties of system parameters as a maximin optimization problem. We show that the adversarial attack obtained by our approach is close to the best adversarial attack found in the case where the system parameters of the MLS are known. Numerical results also show small amount of information of the MLS is enough for an attacker to successfully attack it. This work is closely related to the problem of signal estimation using a limited number of nonlinear sampling measurements.**

I. Introduction

Due to their outstanding practical performance, machine learning systems (MLS), including those based on deep learning, have found applications in many fields such as computer vision, natural language processing, and signal processing [2], [3]. However, the vulnerability of MLSs reported in [4], [6], [7] raise security concerns about these machine learning systems or AI algorithms. As observed by the research community, an imperceptible change of a benign sample can fool the AI algorithms into making wrong decisions, such as assigning wrong labels to the sample.

To understand the fundamental reasons accounting for the vulnerability of MLSs, researchers have devoted significant efforts to designing different attacking techniques under different conditions. In [4], Goodfellow et al. assumed that both the structure and the system parameters of the MLS are known to the attacker (referred to as a white-box attack), and the authors proposed a fast gradient sign method (FGSM) to generate adversarial samples using a well-trained neural network. In [7], Papernot et al. assumed nothing but the access to some input-output pairs of the original MLS (referred to as black-box attack), i.e., access to labels for some given inputs. By acquiring enough such input-label pairs, they proposed to train a substitute MLS using these oracle input-output query pairs, and then generate adversarial samples for the original MLS by studying the substitute MLS. Other works using similar ideas to those in [4], [7] include [1], [5], [6].

In this paper, we consider a gray-box attack against the MLS through an optimization framework, in the sense that the attacker has access only to the structure information of the MLS, and some input-label query pairs of the MLS. Based on the structure information and a limited number of query results, one can obtain incomplete information about the MLS's system parameters. We model the problem of searching for the optimal adversarial attack under such incomplete information as a maximin optimization problem. This optimization problem tries to search for an adversarial attack which is guaranteed to work even under the worst-case system parameters.

Compared with the attacking algorithms in [4], our method requires less knowledge of the MLS, but still achieves very high rate of successful attacks. Our work is also different from [7]. Instead of using the input-output pairs to train a totally new MLS and then generating adversarial attacks based on the newly trained MLS, we generate the adversary attacks by solving maximin optimization problems which can account for worst-case system parameters with performance guarantees. The adversarial attacks our optimization framework finds without knowing the system parameter can often be very close to the best adversarial attacks one can find with full knowledge of the MLS.

Our experiments verify the effectiveness of our optimization framework for designing adversarial attacks. As we increase the number of query samples from the MLS, the adversarial attack that we obtain is increasingly close to the optimal adversarial attack one can obtain with all system parameters known. Very often a small number of queries are enough to design adversarial attacks which can fool the MLS. We further extend our optimization framework from affine multiclass classifiers to designing adversarial attacks against general nonlinear MLS including neural network based classifiers, without knowing the system parameters.

Our results on designing adversarial attacks based on query results are closely related to signal estimations using nonlinear quantized measurements. One can think of the unknown system parameters associated with the MLS as the signals to be estimated, and the query results from the MLS are quantized nonlinear measurements of these system parameters. Instead of directly estimating these system parameters, the ultimate goal of this research is to generate effective adversarial attacks based on the quantized nonlinear measurements of these system parameters.

The rest of this paper is organized as follows. We formally introduce our problem model in Section II. The optimization strategies for attacking affine MLS, and general MLS are presented in Sections III and IV. We present numerical results to validate our strategy in Section V.

II. Problem Statement

Suppose we have an MLS for classification, and for a given sample $x \in \mathbb{R}^d$, we denote the class to which x belongs by $c(x) \in \mathcal{K} := \{1, 2, \cdots, L\}$, where L is the total number of classes. The MLS will determine $c(x)$ by solving $c(x) = \arg\max_k y_k(x; \theta_k)$, where $y_k(x; \theta_k)$ is the score for assigning x to class k, and θ_k is the parameter characterizing $y_k(x; \theta_k)$.

Suppose that we have a benign input sample $x \in \mathbb{R}^d$ which is correctly classified by the MLS, and its true class label is $c(x)$, i.e.,

$$y_{c(x)}(x; \theta_{c(x)}) > y_k(x; \theta_k), \forall k \in \{1, 2, \cdots, L\}, k \neq c(x).$$

We observe a number of input samples and their corresponding output labels, namely the label pairs $(x_i, c(x_i))$, i.e., $c\left(x_i^{(1)}\right) = 1, \forall i \in \mathcal{S}_1 := \{1, \cdots, i_1\}, \cdots, c\left(x_i^{(L)}\right) = L, \forall i \in \mathcal{S}_L := \{1, \cdots, i_L\}$, where $x_i^{(k)}$ is the i-th sample which is classified by the MLS as class k.

Without knowing the system parameters θ_k of the MLS, we want to find a small (so that the perturbation is less detectable) perturbation $\delta \in \mathbb{R}^d$ for x such that the classifier will misclassify it as $c(x + \delta) \neq c(x)$. While our framework also works for the case where we do not know the correct class $c(x)$, to simplify presentations, we assume that we know $c(x)$. We would also like the attack to work under the worst-case scenarios of MLS system parameters.

Mathematically, we propose to solve the following maximin optimization problem:

$$\begin{aligned}
\max_{\delta} \min_{\theta_1, \cdots, \theta_L} \; & y_{c_{ad}}(x + \delta) - y_c(x + \delta), \\
\text{s.t. } & y_1(x_i^{(1)}; \theta_1) > y_k(x_i^{(1)}; \theta_k), \forall i \in \mathcal{S}_1, \forall k \in \mathcal{K} \setminus \{1\}, \\
& \vdots \\
& y_L(x_i^{(L)}; \theta_L) > y_k(x_i^{(L)}; \theta_k), \forall i \in \mathcal{S}_L, \forall k \in \mathcal{K} \setminus \{L\}, \\
& y_c(x; \theta_c) > y_k(x; \theta_k), k \in \mathcal{K} \setminus \{c\}, \\
& \|\delta\| \leq \epsilon,
\end{aligned}$$

where $c(x + \delta)$ is the target class after attacking and $\|\cdot\|$ represents ℓ_2 norm. We also denote $c(x)$ by c, and $c(x + \delta)$ by c_{ad}.

III. Attacking Machine Learning Systems Employing Affine Classifiers

In this section, we consider a machine learning system employing affine classifiers, and transform the maximin optimization problem above to a convex optimization in finding an optimal perturbation $\delta \in \mathbb{R}^d$ without knowing the parameters θ_k of the MLS.

We define the score function for the k-th class as $y_k(x; \theta_k) = w_k^T x + b_k \in \mathbb{R}, k \in \mathcal{K}$, where θ_k contains $w_k \in \mathbb{R}^d$ and $b_k \in \mathbb{R}$. By defining $X^{(k)} = \left[x_1^{(k)} \; x_2^{(k)} \; \cdots \; x_{i_k}^{(k)}\right] \in \mathbb{R}^{d \times i_k}, k \in \{1, 2, \cdots, L\}$, the observation constraint $y_1\left(x_i^{(1)}\right) > y_k\left(x_i^{(1)}\right), \forall i \in \mathcal{S}_1, k \in \mathcal{K} \setminus \{1\}$, now becomes $(w_k - w_1)^T X^{(1)} + (b_k - b_1)\mathbf{1} < \mathbf{0}, k \in \mathcal{K} \setminus \{1\}$, where $\mathbf{1}$ denotes an vector or a matrix with all entries being 1, and $\mathbf{0}$ denotes an vector or a matrix with all entries being 0. The dimensions of $\mathbf{0}$ and $\mathbf{1}$ depend on the context. Similarly, we have $(w_k - w_L)^T X^{(L)} + (b_k - b_L)\mathbf{1} < \mathbf{0}, k \in \mathcal{K} \setminus \{L\}$, and $(w_k - w_c)^T x + (b_k - b_c) < 0, k \in \mathcal{K} \setminus \{c\}$.

We can transform the maximin problem above as

$$\begin{aligned}
\max_{\delta} \min_{w_k, b_k, k \in \mathcal{K}} \; & (w_{c_{ad}} - w_c)^T (x + \delta) + (b_{c_{ad}} - b_c), \\
\text{s.t. } & (w_k - w_1)^T X^{(1)} + (b_k - b_1)\mathbf{1} < \mathbf{0}, \forall k \in \mathcal{K} \setminus \{1\}, \\
& \vdots \\
& (w_k - w_L)^T X^{(L)} + (b_k - b_L)\mathbf{1} < \mathbf{0}, \forall k \in \mathcal{K} \setminus \{L\}, \\
& (w_k - w_c)^T x + (b_k - b_c) < 0, k \in \mathcal{K} \setminus c, \\
& \|\delta\| \leq \epsilon. \qquad (1)
\end{aligned}$$

To solve (1), we transform it to a nested optimization problem. More specifically, (1) can be rewritten as

$$\max_{\delta} f(w_k^*, b_k^*, \delta), \text{ s.t. } \|\delta\| \leq \epsilon, \qquad (2)$$

where $f(w_k^*, b_k^*, \delta)$ is the optimal objective function value of

$$\begin{aligned}
\min_{w_k, b_k, k \in \mathcal{K}} \; & (w_{c_{ad}} - w_c)^T (x + \delta) + (b_{c_{ad}} - b_c), \\
\text{s.t. } & (w_k - w_1)^T X^{(1)} + (b_k - b_1)\mathbf{1} < \mathbf{0}, \forall k \in \mathcal{K} \setminus \{1\}, \\
& \vdots \\
& (w_k - w_L)^T X^{(1)} + (b_k - b_L)\mathbf{1} < \mathbf{0}, \forall k \in \mathcal{K} \setminus \{L\}, \\
& (w_k - w_c)^T x + (b_k - b_c) < 0, k \in \mathcal{K} \setminus c. \qquad (3)
\end{aligned}$$

For (3) , we see that the w_k's and b_k's that satisfy its constraints form a cone. This means if w_k, b_k is in the feasible set, then the tw_k, tb_k will also be in the feasible set for any $t > 0$. Then the minimization will always choose $t \to \infty$ if the objective value can be made negative. To avoid this, we can use a constant to fix the bias difference, i.e., replacing $b_{c_{ad}} - b_c$ by $C = 1$ or -1 (in fact C can be preset as any value).

Let us consider (3) with $b_{c_{ad}} - b_c$ replaced by C. Then the Lagrange dual problem of (3) is given by

$$\begin{aligned}
\max_{\lambda_k^{(i)}, \alpha_k^{(c)}} \; & \left(C + C(-1)\mathbf{1}\lambda_c^{c_{ad}} + C\mathbf{1}\lambda_{c_{ad}}^{(c)} + C\alpha_{c_{ad}}^{(c)}\right), \\
\text{s.t. } & \lambda_k^{(1)} \geq \mathbf{0} \in \mathbb{R}^{i_1}, k \in \mathcal{K} \setminus \{1\}, \\
& \vdots \\
& \lambda_k^{(L)} \geq \mathbf{0} \in \mathbb{R}^{i_L}, k \in \mathcal{K} \setminus \{L\}, \\
& \alpha_k^{(c)} \geq 0 \in \mathbb{R}, k \in \mathcal{K} \setminus \{c\}, \\
& C_j^{(w)} = \mathbf{0} \in \mathbb{R}^d, C_j^{(b)} = 0 \in \mathbb{R}, j \in \mathcal{K}, \qquad (4)
\end{aligned}$$

where $C_j^{(w)}$ and $C_j^{(b)}$ are defined in (5) and (6):

$$\begin{aligned}
C_c^{(w)} &= -(x+\delta) + \sum_{k\neq c}\left(X^{(k)}\lambda_c^{(k)} - X^{(c)}\lambda_k^{(c)} - x\alpha_k^{(c)}\right),\\
C_{c_{ad}}^{(w)} &= (x+\delta) + \sum_{k\neq c_{ad}}\left(X^{(k)}\lambda_{c_{ad}}^{(k)} - X^{(c_{ad})}\lambda_k^{(c_{ad})}\right) + x\alpha_{c_{ad}}^{(c)},\\
C_j^{(w)} &= \sum_{k\neq j}\left(X^{(k)}\lambda_j^{(k)} - X^{(j)}\lambda_k^{(j)}\right) + x\alpha_j^{(c)}, j\neq c, c_{ad}, \quad (5)
\end{aligned}$$

and

$$C_j^{(b)} = \begin{cases} \sum_{k\in\mathcal{K}\setminus\{c,c_{ad}\}}\left(\mathbf{1}\lambda_c^{(k)} - \mathbf{1}\lambda_k^{(c)}\right), j = c,\\ \sum_{k\in\mathcal{K}\setminus\{c,c_{ad}\}}\left(\mathbf{1}\lambda_{cad}^{(k)} - \mathbf{1}\lambda_k^{(c_{ad})}\right), j = c_{ad},\\ \sum_{k\in\mathcal{K}\setminus\{j\}}\left(\mathbf{1}\lambda_j^{(k)} - \mathbf{1}\lambda_k^{(j)}\right), j\neq c, c_{ad}, \end{cases} \quad (6)$$

where $\mathbf{1}\in\mathbb{R}^{1\times i_k}$ if it is associated with class k.

Combining (2) and (4), we can transform the nested optimization problem to

$$\begin{aligned}
\max_{\delta}\ &\max_{\lambda_k^{(i)},\alpha_k^{(c)}} \left(C + C(-1)\mathbf{1}\lambda_c^{c_{ad}} + C\mathbf{1}\lambda_{c_{ad}}^{(c)} + C\alpha_{c_{ad}}^{(c)}\right),\\
\text{s.t. } &\|\delta\|\leq\epsilon,\\
&\lambda_k^{(1)} \geq \mathbf{0}\in\mathbb{R}^{i_1}, k\in\mathcal{K}\setminus\{1\},\\
&\vdots\\
&\lambda_k^{(L)} \geq \mathbf{0}\in\mathbb{R}^{i_L}, k\in\mathcal{K}\setminus\{L\},\\
&\alpha_k^{(c)} \geq 0\in\mathbb{R}, k\in\mathcal{K}\setminus\{c\},\\
&C_j^{(w)} = \mathbf{0}\in\mathbb{R}^d, C_j^{(b)} = 0\in\mathbb{R}, j\in\mathcal{K}, \quad (7)
\end{aligned}$$

which is actually a convex optimization and can be solved efficiently. Once we have obtained the solution for this optimization problem, i.e., δ^*, we can plug it into (1) and solve the following optimization problem to estimate w_k and b_k,

$$\begin{aligned}
\min_{w_k,b_k,k\in\mathcal{K}} &(w_{c_{ad}} - w_c)^T(x+\delta^*) + (b_{c_{ad}} - b_c),\\
\text{s.t. } &(w_k - w_1)^T X^{(1)} + (b_k - b_1)\mathbf{1} < \mathbf{0}, \forall k\in\mathcal{K}\setminus\{1\},\\
&\vdots\\
&(w_k - w_L)^T X^{(L)} + (b_k - b_L)\mathbf{1} < \mathbf{0}, \forall k\in\mathcal{K}\setminus\{L\},\\
&(w_k - w_c)^T x + (b_k - b_c) < 0, k\in\mathcal{K}\setminus c(x). \quad (8)
\end{aligned}$$

IV. Attacking Machine Learning Systems Employing General Classifiers

In this section, we formulate the problem of attacking machine learning systems employing general classifiers beyond affine classifiers. Specifically, we consider a classifier based on a neural network. At the last layer of the neural network, there are L neurons, which output the scores for each of the L classes. The classifier will select the class which has the largest score (for example, the MNIST classifier [8]).

Suppose we have a score function $f:\mathbb{R}^d\to\mathbb{R}^L$ which we compute using a s-layer nonlinear neural network, i.e., $f(x)$ is equal to

$$a^{(s)}\left(W^{(s)}a^{(s-1)}\left(\cdots a^{(1)}\left(W^{(1)}x + b^{(1)}\right)\cdots\right) + b^{(s)}\right),$$

where $a^{(j)}$ is the element-wise activation function in the j-th layer, $W^{(j)}\in\mathbb{R}^{d_j\times d_{j-1}}$ and $b^{(j)}\in\mathbb{R}^{d_j}$ are the weight matrix and bias vector in the j-th layer. Here $d_0 = d$ and $d_s = L$, and we omit the $W^{(j)}$ and $b^{(j)}$ for convenience of presentation.

Then we can model designing the most effective attack as

$$\begin{aligned}
\max_{\delta}\ &\min_{W^{(j)},b^{(j)}} [f(x+\delta)]_{c_{ad}} - [f(x+\delta)]_c,\\
\text{s.t. } &\|\delta\|_2\leq\epsilon,\\
&[f(x_i^{(1)})]_1 \geq [f(x_i^{(1)})]_k, i\in\mathcal{S}_1, k\in\mathcal{K}\setminus\{1\},\\
&[f(x_i^{(2)})]_2 \geq [f(x_i^{(2)})]_k, i\in\mathcal{S}_2, k\in\mathcal{K}\setminus\{2\},\\
&\vdots\\
&[f(x_i^{(L)})]_L \geq [f(x_i^{(L)})]_k, i\in\mathcal{S}_L, k\in\mathcal{K}\setminus\{L\},\\
&[f(x)]_c - [f(x)]_k \geq 0, k\in\mathcal{K}\setminus\{c\},\\
&[f(x+\delta)]_{c_{ad}} - [f(x+\delta)]_k \geq 0, k\in\mathcal{K}\setminus\{c_{ad}\}, \quad (9)
\end{aligned}$$

where $[f(x)]_k$ is the k-th element of $f(x)$. Note that here we impose one extra constraint to force the system to classify the perturbed sample as class c_{ad}.

V. Numerical Results

In this section, we present numerical results to evaluate the optimization-based attack strategy. To better illustrate the idea, we consider the adversarial attack problem in a low dimensional space, i.e., an affine three-class classifier using 2 features. Without loss of generality, we assume the benign sample $x\in\mathbb{R}^2$ is in class $c = 2$, and let $c_{ad} = 1$. It is worth pointing out that the adversarial sample $x+\delta$ may not be guaranteed to be in class c_{ad}, but it is expected to be in a class different from $c = 2$. The perturbation δ_{un}^* found with unknown w_k, b_k will be obtained by solving (7), and the perturbation δ^* with w_k, b_k known is obtained by solving (10),

$$\begin{aligned}
\max_{\delta}\quad &(w_1 - w_2)^T(x+\delta) + (b_1 - b_2),\\
\text{s.t.}\quad &\|\delta\|\leq\epsilon. \quad (10)
\end{aligned}$$

In our first experiment, we want to see the relation between δ_{un}^* and δ^*. We design a classifier with parameters $w_1 = [-0.65, -1.11]^T, b_1 = -0.56, w_2 = [1.18, -0.85]^T, b_2 = 0.18, w_3 = [-0.76, -0.57]^T, b_3 = -0.20$. Then we generate 1000 inputs (x) for the classifier, and elements of x are i.i.d. following the standard Gaussian distribution. For the 1000 input samples, the classifier will report their corresponding labels. The tolerance on the attack perturbation magnitude is set to be $\epsilon = 2.00$. We use (7) to get δ_{un}^* and (10) to get δ^*. Finally the adversarial samples obtained are $x+\delta_{un}^*$ and $x+\delta^*$, where the benign sample x is taken from class 2, i.e., $[2.15\ \ -0.63]^T$. The results are shown in Figure 1. From the results, we can see the adversarial attacks based on only query results and the adversarial attacks constructed based on actual system parameters are very close. In fact $\frac{\|\delta_{un}^* - \delta^*\|}{\|\delta^*\|} = 0.12$ in Figure 1. We expect that as we increase the number of sensing samples, we can get smaller $\frac{\|\delta_{un}^* - \delta^*\|}{\|\delta^*\|}$ on average.

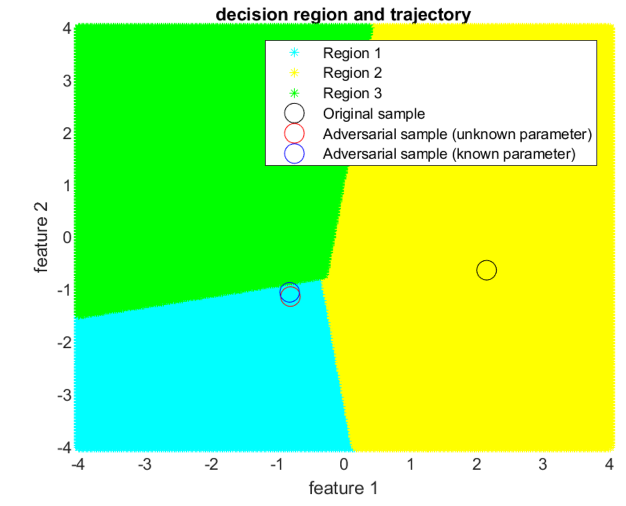

Figure 1: Decision regions and adversarial samples $x + \delta$.

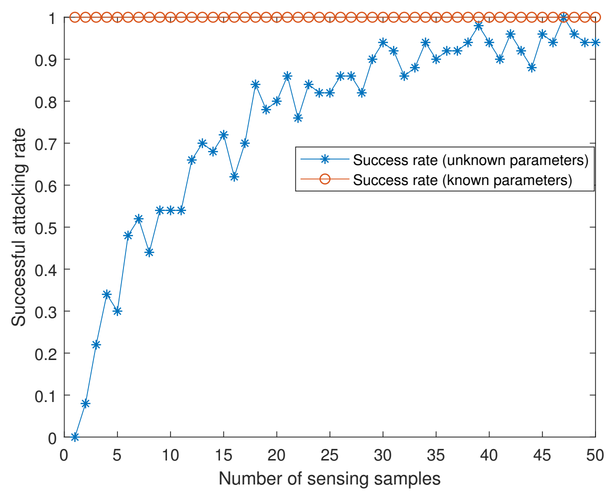

Figure 3: Effect of the number of measurements on the rate of successful attack.

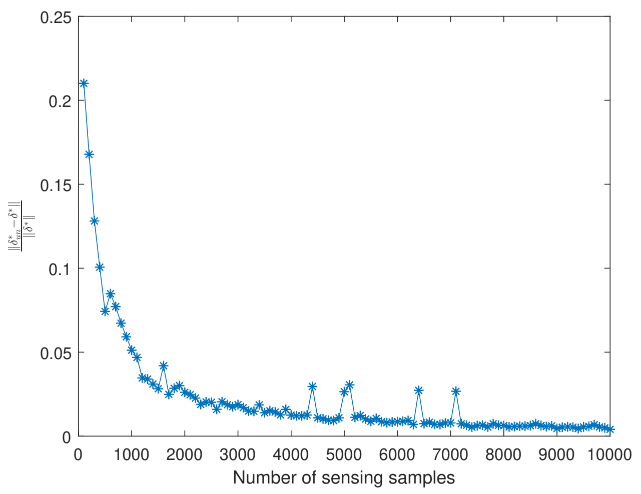

Figure 2: Effect of the number of query pairs on perturbation estimation. For each choice of number of query pairs, in total 50 trials are performed and the average is computed.

Next, we conduct another experiment to evaluate how $\frac{\|\delta^*_{un}-\delta^*\|}{\|\delta^*\|}$ will change with respect to the number of query pairs $(x, c(x))$. We use the same setup described in the previous experiment. For each choice of the number of sensing samples, we perform the experiment for 50 instances and get the corresponding $\frac{\|\delta^*_{un}-\delta^*\|}{\|\delta^*\|}$. Then we take the average of the 50 trials. The result is presented in Figure 2. As shown in the figure, the perturbation found without knowing the system parameters gets increasingly close to the one found when the system parameters is known.

Finally, we evaluate the rate of successful attack, which is explained below, based only on a small number of input-output query pairs of the MLS. We use an affine multi-class classifier with the same system parameters as those in the previous experiments. We choose $\epsilon = 3.00$. We experiment with different numbers of query pairs. For each choice of number of query pairs, we randomly generate 50 benign samples within class 2. We use (7) and (10) to find the perturbation δ^*_{un} and δ^*, namely the attacks constructed based only on query pairs and attacks constructed while fully knowing the parameters. For each of the 50 instances in class 2, if the adversarial attack fools the classifier into a different class, i.e., class 1 or class 3, then we say it is a successful attack. The rate of successful attack is computed by the number of successful attacks divided by 50. The results are presented in Figure 3. From the results, we can see a very small number of query pairs are often enough for acquiring information of the classifier to design effective effective attack.

References

[1] N. Carlini and D. Wagner. Adversarial examples are not easily detected: bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, AISec '17, pages 3–14, New York, NY, USA, 2017. ACM.

[2] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[3] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[4] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv:1412.6572 [cs, stat]*, December 2014. arXiv: 1412.6572.

[5] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv:1607.02533 [cs.CV]*, July 2016.

[6] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. DeepFool: a simple and accurate method to fool deep neural networks. pages 2574–2582, 2016.

[7] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. Celik, and A. Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, pages 506–519, New York, NY, USA, 2017. ACM.

[8] P. Zadeh, R. Hosseini, and S. Sra. Deep-RBF networks revisited: robust classification with rejection. *arXiv:1812.03190 [cs, stat]*, December 2018.