DyAdvDefender: An Instance-based Online Machine Learning Model for Perturbation-trial-based Black-box Adversarial Defense

Miles Q. Li[a], Benjamin C. M. Fung[b,*] and Philippe Charland[c]

[a]*School of Computer Science, McGill University, Montreal, Canada*
[b]*School of Information Studies, McGill University, Montreal, Canada*
[c]*Mission Critical Cyber Security Section, Defence R&D Canada, Quebec, Canada*

ARTICLE INFO

Keywords:
Adversarial defense, black-box attack, online machine learning

ABSTRACT

Recent research indicates that machine learning models are vulnerable to adversarial samples that are slightly perturbed versions of natural samples. Adversarial samples can be crafted in white-box or black-box scenario. In the black-box scenario adversaries possess no knowledge of the detailed architecture and parameters of the model they attack, and they seek information by querying the model with multiple, slightly perturbed samples to achieve their attack purpose. The difficulty in recognizing adversarial samples arises from the fact that the perturbations are often imperceptible, yet effective in misleading machine learning models. Existing defense methods are static, and they cannot dynamically evolve to adapt to adversarial attacks, which unnecessarily disadvantages them. In this paper we propose a novel dynamic defense method called *DyAdvDefender*, and we show that a dynamic defense method can effectively utilize previous experience to defend against black-box attacks. Extensive experimental results suggest that DyAdvDefender outperforms existing static methods in terms of defense effectiveness while keeping the original classification accuracy with only limited extra time consumption.

1. Introduction

Machine learning models, especially deep neural networks, have achieved state-of-the-art performance in many fields in the cyber world, e.g., image classification, malware detection, and natural language processing [7, 24, 46]. However, machine learning models can be compromised by adversarial attacks [18, 23, 31, 50] that intend to cause the machine learning models to misclassify samples that contain often imperceptible, but carefully selected perturbations. This vulnerability can be exploited to induce catastrophic effect: an autonomous vehicle that fails to recognize a stop sign may cause a traffic accident; failure to recognize a malware program may cause an enormous loss to a computer user. Evidence proves that adversarial attacks have come to real-world scenarios [37, 23][1]. As a result, adversarial attack and defense are topics extensively studied [44].

A machine learning model is a function $f(\boldsymbol{x})$ that maps an input sample $\boldsymbol{x}$ to an output $\boldsymbol{y}$, an n-dimensional vector, where n is the number of classes. Each element of $\boldsymbol{y}$ corresponds to the probability that $\boldsymbol{x}$ belongs to a certain class. The objective of an adversary is to add a minimal perturbation $\boldsymbol{\delta}$ to a natural sample $\boldsymbol{x}$ so that $f(\boldsymbol{x}+\boldsymbol{\delta})$ presents a false result. Adversarial samples with large perturbations are relatively easy to detect; however, adversarial attacks present a challenge because the perturbation added to a natural sample is usually very small, yet very effective [9, 44]. Attacks can be conducted in two different scenarios. One is the white-box attack scenario in which adversaries possess the whole formula of f, and therefore they can utilize the gradient that shows the most efficient perturbation to perform adversarial attacks [18, 23, 31]. Thus, the differentiable property of deep learning-based systems makes them especially vulnerable to white-box adversarial attacks [27, 9, 15]. The second is the black-box attack scenario in which the adversary has no knowledge of the target deep learning model but can request its output for any valid input. This is more common because most classification service providers do not reveal the details of their systems. There are two ways to conduct an attack. One is to train a substitute model and craft adversarial samples by attacking the substitute model, as in the white-box setting, and then use the adversarial samples to attack the target model [30]. However, this kind of attack is ineffective when the substitute and target models have little in common, and it has been well defended

*Corresponding author.
miles.qi.li@mail.mcgill.ca (M.Q. Li); ben.fung@mcgill.ca (B.C.M. Fung); philippe.charland@drdc-rddc.gc.ca (P. Charland)
ORCID(s): 0000-0001-8423-2906 (B.C.M. Fung)
[1]https://www.youtube.com/watch?v=zQ_uMenoBCk

by existing methods [29, 10, 21, 19]. The other kind of attack is to estimate the gradient or derivative of the target model by examining the relation between changes of $\boldsymbol{y}$ and the perturbations of $\boldsymbol{x}$. This kind of attack can bypass existing defense mechanisms, and thus it is the focus of this paper.

As a principal step to build robust systems [9, 33, 35, 39], defense against adversarial attacks can be broadly categorized as proactive or reactive methods [41]. The chosen method can either fortify the robustness of the machine learning model per se with embedded mechanics or special training procedures [11, 13, 20, 32, 34, 41], or it can defend the target model by recognizing adversarial samples [16, 19, 25, 45]. A common issue with existing defense methods is that they are static, and therefore they cannot update their states while countering adversaries. In the black-box scenario, a successful attack requires more than thousands of queries [21]. Conversely, each query could also be used by the system to counter the attacks. Existing methods do not have the mechanics in place to utilize them, which leads to an unnecessary disadvantage of the defenders because adversaries can update their attack strategies according to the information acquired from their pry attempts. A more effective defense method that can dynamically update its state is yet to be proposed.

Online machine learning is a family of machine learning in which a model updates its state whenever it receives a new query sample, as opposed to offline learning, in which a model is trained on a fixed training set and then remains static. The advantage of online machine learning is the adaptive capability that allows it to update its state according to the new experience. In this paper, we propose *DyAdvDefender*, an instance-based online machine learning model to defend against black-box adversarial attacks. DyAdvDefender can recognize a perturbed sample that originates from the same sample as a previously queried sample, and it can output the same classification result for all the perturbed samples that have the same origin. Thus, the classification result is intact, and the adversaries cannot estimate the gradient or derivative, as the perturbations of inputs cause no change of the output. To implement this approach we propose solutions to address two main challenges: (i) how to determine whether two samples have the same origin and (ii) how to keep the time consumption of the defense mechanism in a controllable range.

We summarize the contributions of this paper as follows:

1. We propose the **first** defense method that is based on **online machine learning** and **instance-based learning** to dynamically update its states according to received attacks.

2. We propose a novel dedicated locality sensitive hashing (LSH)-based optimization method for DyAdvDefender to index and retrieve samples with an exemplary efficiency sufficient for real-world applications.

3. We propose a new evaluation procedure to assess an online defense method because all previous evaluation procedures are only suitable for static defense methods.

4. Extensive experiments on different types of datasets suggest that our defense method, DyAdvDefender, significantly outperforms existing state-of-the-art defense methods in terms of defense effectiveness, while not compromising classification accuracy on natural samples.

2. Related Work

2.1. Adversarial Attack

2.1.1. White-box Adversarial Attack

In the white-box adversarial attack setting, an adversary uses the formula of the classification model to perform the attack. Goodfellow et al. [18] propose the fast gradient sign method (FGSM) to craft adversarial samples. They perturb each sample only once towards the gradient sign direction with a small step. Kurakin et al. [23] propose the iterative gradient sign (IGS) method to craft adversarial samples through multiple rounds. In each round, IGS perturbs the pixels towards the gradient sign direction and clips the perturbation using a small value. Papernot et al. [31] propose the forward derivative method (FDM). It evaluates the sensitivity of the output to each feature using the Jacobian matrix of the model, and then it constructs an adversarial saliency map based on the Jacobian matrix indicating which input features should be included in each iteration of perturbation until it succeeds. Moosavi-Dezfooli et al. [27] propose the Deepfool algorithm, which attacks a system by pushing a sample towards the orthogonal vector of the target classification boundary. Carlini and Wagner [8] take the adversarial attack as an optimization problem in which the objective function includes the distance between the adversarial sample and its original sample as well as the difference between the probabilities of the target class and the real class. Other studies use optimization-based methods with different settings on the objective function, distance measure, and constraints [37, 36, 26]. Furthermore, generative networks are also adopted to synthesize adversarial samples [47].

2.1.2. Black-box Adversarial Attack

In the black-box setting attacks are conducted based on the information acquired by querying the target classification system. The following are the two principal types of black-box attack methods.

Substitute Model Papernot et al. [30] expect that adversarial samples have transferability, indicating that a sample that compromises one model is likely to compromise another. Hence, they propose to obtain the outputs corresponding to a set of synthetic samples from the target model and use them to train a substitute model. They craft adversarial samples by attacking the substitute model in the white-box setting and expect that those samples can mislead the target model. Recent works [29, 29, 10, 21, 26, 28] show that the transferability of adversarial examples crafted from a substitute model is limited, and it can be defended effectively, especially when the substitute model is complex or quite different from the target model. Additionally, this kind of attack is already well defended by existing methods [19].

Perturbation Trial A more effective kind of black-box attack method is based on perturbation trials. Because the output of the target model can be acquired by the adversaries, they can attempt to perturb certain features by increasing and decreasing the values and check whether either would lead the output towards their objective direction, e.g., the probability of the target class increases. Essentially, instead of using the analytical solution to acquire the gradient in the white-box setting, this solution for a black-box setting uses numerical differentiation to achieve the same purpose. This technique is also widely used in gradient checking for validating the implementation of back propagation for training neural networks [42]. The adversary repeats that procedure multiple times and accumulates effective perturbations until an adversarial sample is found or a maximum number of trials has been performed. Studies [10, 40, 21] following this methodology use different strategies to improve the query efficiency. Existing defense mechanisms cannot yet handle this type of attacks. Therefore, our proposed method aims at defending against such attacks.

2.2. Adversarial Defense

2.2.1. Proactive Defense

With proactive defense methods, machine learning models per se are fortified to be robust against adversarial attacks, and there is no independent defense component outside the machine learning model. The defense is achieved by modifying the input to cancel out the effects caused by the perturbation [43], incorporating some components in the machine learning model [11, 13, 49], or encompassing the robustness in the training objectives or procedures [20, 48, 32, 34, 41, 26, 38, 28]. Studies show that those defense methods are not effective in the white-box attack scenario where adversaries know those mechanisms and cannot defend against high-confidence adversarial examples or Jacobian matrix-based attacks [8, 41].

2.2.2. Reactive Defense

Reactive methods are based on recognizing adversarial samples. Feinman et al. [16] propose to use kernel density (KD) estimation as a measure to differentiate adversarial samples from natural samples. Grosse et al. [19] add a new class for adversarial samples to the K existing classes and train a machine learning model to classify a sample to one of the $K + 1$ classes. Lu et al. [25] propose to apply the RBF-SVM model on activation patterns as its features to detect adversarial attacks. Following a different logic, Xu et al. [45] apply deep neural networks on two kinds of features: squeezed and unsqueezed features. They find that the predictions with these two kinds of features are quite different on adversarial samples, which is not the case for natural samples. Therefore, they propose to check the difference between predictions on squeezed and unsqueezed features to identify adversarial samples. Those methods could defend against substitute model attacks in their experiments, but their performance on defending perturbation trial attacks is limited [4].

3. Proposed Defense Method

In this section, we describe the proposed defense method: DyAdvDefender. As explained earlier, substitute model attacks and perturbation trial attacks are different methodologies in the black-box attack scenario, and the former is a lesser threat because of its limitations and the effectiveness of existing defense mechanisms against it. We therefore focus on defending against perturbation trial-based black-box attacks.

3.1. Preliminaries

There are slightly different settings for black-box attacks. We follow the setting [10, 21, 40] in which the adversaries have no knowledge of the attacked system but have access to the complete output, including the probability that a query sample belongs to each class.

To craft an adversarial sample with perturbation trial attacks, the adversaries perturb a natural sample many times to form **prying samples** that are used to query the target model. They accumulate the effective perturbations until a successful **adversarial sample** is found. This implies that during the attack, the target model receives several prying samples that have the same **origin**, which is the natural sample. We give a formal definition of a sample's *origin* as follows.

Definition 1 (Origin)**.** *Consider a natural sample $\boldsymbol{x_0}$, for any sample $\boldsymbol{x} = \boldsymbol{x_0} + \boldsymbol{\delta}$, that is crafted by perturbing $\boldsymbol{x_0}$. $\boldsymbol{x_0}$ is called the origin of $\boldsymbol{x}$.*

Due to the nature of adversarial samples and their crafting, the perturbations of adversarial samples to their original samples should be small ($||\boldsymbol{\delta}|| \ll ||\boldsymbol{x_0}||$) for multiple reasons [37]. For example, for images, perturbations that are smaller than 1.0E-2 per pixel in L_2 measure are small enough and also sufficient to craft adversarial samples [40]. The adversaries want them to be hardly perceptible for the purposes of stealthiness and invariant semantics. From the perspective of calculus, the perturbation should be small enough so that the first-order gradient could be approximated, and thus the accumulation of the perturbations could stay effective and efficient. This property of small perturbations can be used to determine whether two samples have the same origin.

3.2. Overall Defense Mechanism

We present the unoptimized DyAdvDefender in Algorithm 1. When the system receives a query, it checks whether there is any indexed sample that has the same origin. If so, it outputs the prediction of the indexed sample and indexes the query with the output. If there is no indexed sample that has the same origin, it outputs the predicted result of the query by the machine learning model to defend, and it indexes the input and output pair as well.

Algorithm 1: Unoptimized DyAdvDefender.

```
Data: a query $\boldsymbol{x}$, a set of indexed samples $S$, the machine learning model to defend $C$
Result: Classification result $y$
for $(\boldsymbol{x_0}, \boldsymbol{y_0}) \in S$ do
    if $Same\_Origin(\boldsymbol{x_0}, \boldsymbol{x})$ is True then
        $\boldsymbol{y} = \boldsymbol{y_0}$;
        $Index((\boldsymbol{x}, \boldsymbol{y}), S)$;
        Return $\boldsymbol{y}$;
    end
end
$\boldsymbol{y} = C(\boldsymbol{x})$;
$Index((\boldsymbol{x}, \boldsymbol{y}), S)$;
Return $\boldsymbol{y}$;
```

We have two assertions about this system if the $Same\_Origin$ sub-algorithm performs perfectly:

1. It outputs the correct classification result if the defended model can correctly predict the origin of the query sample.

2. It does not reveal any gradient information.

We have the first assertion because the outputs for all samples that have the same origin are the result of the original sample or a sample that is randomly perturbed once; it is unlikely that one random perturbation can mislead the classifier [10, 40, 21].

We have the second assertion because the output is not affected by the perturbation on the input. Thus, the adversaries cannot estimate the gradient or the partial derivatives with respect to input variables in the black-box setting.

DyAdvDefender is an online machine learning model because its state changes as more queries are received. It is also an instance-based learning model because the prediction depends on received queries. To the best of our knowledge, this is the first defense method that is based on online machine learning or instance-based learning. Even though the idea is simple, there are two challenges to address: 1) how to determine whether two samples have the same origin, and 2) how to keep the classification procedure efficient even when a large number of samples are indexed.

3.3. Determination of Same Origin

Based on the property that the perturbations in prying and adversarial samples are small, we propose to recognize samples having the same origin based on their distance. Let us imagine an *ideal* distance threshold θ_0 that can be used to determine the distance between two samples as follows.

Definition 2 (Ideal Distance Threshold θ_0)**.** *Let D be a distance measure. Consider two random samples $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$. Let θ_0 be a specific threshold defined on measure D such that if the distance between any two samples on measure D is smaller than θ_0, they have the same origin. If the distance is no less than θ_0, they have different origins. It can be expressed as follows:*

$$Same\_Origin(\boldsymbol{x_1}, \boldsymbol{x_2}) = \begin{cases} True & D(\boldsymbol{x_1}, \boldsymbol{x_2}) < \theta_0 \\ False & D(\boldsymbol{x_1}, \boldsymbol{x_2}) \geq \theta_0 \end{cases}$$

The θ_0 is *ideal*. It is unknown whether it exists. However, we can obtain an empirical approximation of θ_0. Let us discuss the measure of distance first.

3.3.1. Measure of Distance

The best measure for computing the distance between two samples is determined by the nature of the field of application. For samples that can be represented as a feature vector of a fixed length, any L_p Minkowski distance could be a valid measure. If the vectors are real-valued (e.g., the greyscale values of pixels of an image), the Euclidean distance (L_2): $D(\boldsymbol{x_1}, \boldsymbol{x_2}) = ||\boldsymbol{x_1} - \boldsymbol{x_2}||_2$ is the natural choice to measure the distance of two vectors. If different features are at different scales, feature standardization should be applied on the feature vectors before computing the distance so that the value of each dimension ranges between -0.5 and 0.5. If the features represented by the vectors are integers (e.g., the frequencies of different printable strings in an executable) or binary (e.g., whether a DLL function is imported in an executable), the Manhattan distance (L_1) is the natural choice.

In the fields where samples cannot be represented as a vector but rather as a sequence, the way to measure the distance of two samples is not obvious and is another research topic. We do not elaborate on this because it is not the focus of this paper.

3.3.2. Threshold of Distance

We have defined θ_0, the *ideal* distance boundary, to determine whether two samples have the same origin. Any two samples that have a distance no less than θ_0 have different origins. Conversely, any two samples that have a distance less than θ_0 have the same origin. There is no theory to indicate whether such an ideal threshold exists or how to find it. The whole concept is similar to training an artificial neural network: we suppose the input and output have an intrinsic relation described as a function of a specific neural network architecture and unknown parameters. It is unknown whether the function could resemble the underlying relation between the output and input, and it is unknown how to find the precise parameters working for all samples. We can only fit the function onto a finite set of samples (i.e., the training set) and expect it to generalize well to unseen samples that are drawn from the same distribution as the training set. This is called empirical risk minimization (ERM) of the parameters. We also want to use a dataset to empirically approximate θ_0. Let us first identify two corollaries about θ_0.

Corollary 1. *θ_0 is the minimum distance between any two samples that have different origins.*

Corollary 2. *θ_0 is the maximum distance between any two samples that have the same origin.*

θ_0 is defined with respect to the complete set of samples in a field. The empirical approximation of it is computed based on a finite set of samples that are randomly drawn from the complete set. Corollary 2 narrates θ_0 from the perspective of samples that have the same origin. To empirically approximate θ_0 based on this corollary requires a dataset of adversarial samples. We do not have a set of adversarial samples that are drawn from the complete set of adversarial samples. Thus, this corollary is not a good theoretical support to approximate θ_0. Corollary 1 is narrated from the perspective of samples that have different origins, exactly what a training set provides us. Therefore, it can be used to empirically approximate θ_0.

In many applications, e.g., image classification, audio processing, etc., the samples in the training sets are correctly labelled natural samples. Thus, we can get the empirical approximation of θ_0 with a training set as follows. We compute the distance between every two samples in the training set. Following Corollary1, we set θ_0^*, an empirical approximation of θ_0, to be the minimal distance between any two natural samples from any classes in the training set.

In other cases, the training set contains samples that have the same origin. For instance, a training set for malware detection probably contains malware samples from the same family, and they are slightly different versions to compromise detectors. There are also benign executables that have the same origin, e.g., a Dynamically Linked Library (DLL) file that is patched for a vulnerability and its unpatched version have the same origin. Therefore, we cannot compute the minimal distance between any two samples in such a dataset to approximate θ_0. However, as two samples that have the same origin should belong to the same class, we can get a distance threshold that is an inferior approximation of θ_0, but is as equally effective. We set θ_0^* to be the minimal distance between two samples that (1) belong to different classes in the training set and (2) are correctly classified by the machine learning model that we aim to defend. This is a less good approximation of θ_0, because it may recognize two samples that belong to the same class, and have different origins, as having the same origin. It is equally effective due to two reasons. First, it may recognize two samples in the same class that have different origins as having the same origin, and it would not recognize two samples in different classes as having the same origin, thus it would still give the correct prediction. Second, the derivative information is still not revealed to adversaries. We only compare the distances between samples that are correctly predicted by the machine learning model to defend because in these cases, the incorrectly predicted samples contain adversarial samples, abnormal samples, or mislabeled samples that could mislead the computation of θ_0^*.

3.4. Optimizations

We have described how DyAdvDefender handles the input and produces the output. However, the vanilla defense procedure is not efficient enough, because it unnecessarily compares a query sample with all previously queried samples. Below we describe optimization methods to improve on this.

3.4.1. Locality Sensitive Hashing-based Origin Search

When there are a large number of indexed samples, computing the distance between a query sample and all indexed samples is computationally expensive and less practical in real-world application. Locality sensitive hashing (LSH) has been shown to be efficient in finding sets of nearest neighbors in previous works [22, 6]. Therefore, we propose a dedicated LSH-based algorithm that is different from previous works to efficiently retrieve a subset of indexed samples that may contain the samples having the same origin as the query sample.

The LSH function family we use can be described as follows: each hash function contains a random vector $\boldsymbol{r}$ that has the same dimension as the feature vector of a sample. The value of each entry of $\boldsymbol{r}$ is independently drawn from standard Gaussian distribution. The hash value of a sample $\boldsymbol{u}$ is computed as follows:

$$h_{\boldsymbol{r}}(\boldsymbol{u}) = \begin{cases} 1 & \boldsymbol{u} \cdot \boldsymbol{r} > 0 \\ 0 & \boldsymbol{u} \cdot \boldsymbol{r} \leq 0 \end{cases}$$

It has been proven [17, 2] that for vector $\boldsymbol{u}$ and vector $\boldsymbol{v}$, the probability that they have the same hash value with $h_{\boldsymbol{r}}$ is as follows:

$$Pr[h_{\boldsymbol{r}}(\boldsymbol{u}) = h_{\boldsymbol{r}}(\boldsymbol{v})] = 1 - \frac{\theta(\boldsymbol{u}, \boldsymbol{v})}{\pi} \tag{1}$$

Two vectors that have a small angle have a large probability to have the same hash value and vice versa. The angle between a natural sample ($\boldsymbol{x_0}$) and its perturbed version ($\boldsymbol{x} = \boldsymbol{x_0} + \boldsymbol{\delta}$, where $||\boldsymbol{\delta}|| \ll ||\boldsymbol{x_0}||$) is very small, and both tend to have the same hash values.

We use k such LSH functions to compute a signature of k bits for each sample, so there are 2^k possible different signatures. To increase the chance that those k LSH functions could evenly split different samples, we randomly generate LSH functions until we get k of them, of which the hash values of 40%∼60% of training samples (i.e., natural samples) are 1, and the remaining 40%∼60% are 0. We use a normal hash table to store and retrieve the set of samples corresponding to each signature. Samples that have the same origin tend to have the same signature. Given a query sample, we compute its signature and the distance between the query sample and the indexed samples that have the same signature to see if any of them has the same origin as the query sample. When there are n samples indexed, the number of indexed samples a query sample is expected to compare with is on average $n/2^k$, in an ideal situation.

This is exponentially more efficient in terms of k than comparing each query sample with all n indexed samples. For example, to compare each query with 1,024 indexed samples on average, which could be efficiently done by most GPUs for data science nowadays, only 10 LSH functions are required when there are 1 million indexed samples, and 20 LSH functions when there are 1 billion indexed samples.

As a probabilistic algorithm, there is inevitably a small probability that two samples with the same origin have different signatures. Therefore, we use m sets of k LSH functions to compute m signatures. Each indexed sample has m signatures and is put in m sets. We compute m signatures of a query sample and compare it with any sample in the union of the m sample sets corresponding to the m signatures. Empirically, we set $m = 2$ since it is sufficient to allow two samples that have the same origin to be in one set [14]. Algorithm 2 describes the procedure to find the set of samples that potentially have the same origin as a query sample.

Algorithm 2: The function to return a set of indexed samples that may have the same origin as the query sample.

```
Data: a query $\boldsymbol{x}$, $m \times k$ LSH functions $H$, $m$ hash tables that map a signature to a set of samples $HT$
Result: A set of indexed samples that potentially have the same origin as the query sample $S_0$
$S_0 = \emptyset$;
while $i < m$ do
    sig = new List<Bit>();
    while $j < k$ do
        $h_{\boldsymbol{r}} = H(i, j)$;
        $sig.append(h_{\boldsymbol{r}}(\boldsymbol{x}))$;
    end
    $S_0.union(HT_i.get(sig))$;
end
Return $S_0$;
```

3.4.2. *Selective Indexing*

To index every query sample gives DyAdvDefender the greatest opportunity to identify a new query sample as a perturbed version of a previous query sample. However, it is also very inefficient because an adversary may query with tens of thousands of prying samples that have the same origin [21].

Another extreme practice is to only index a query sample if DyAdvDefender cannot find an indexed sample that has the same origin. This setting yields the highest efficiency but could impair the effectiveness. When adversaries realize that perturbation does not work with our defense, they may gradually increase the scale of the perturbation, in which case DyAdvDefender may be compromised.

To keep both effectiveness and efficiency, we index two kinds of query samples. One is query samples with which no indexed samples have the same origin. The other is query samples that have the same origin as some indexed samples, but the distance between the query samples and the closest indexed samples are larger than $\theta_0^*/10$. By indexing the sample where the perturbation is one order of magnitude smaller than θ_0^*, we can invalidate the incremental attacks that may exist in practice as we mentioned above.

3.4.3. *Optimized Defense Mechanism*

Algorithm 3 presents the optimized version of DyAdvDefender that encompasses the LSH-based origin search and selective indexing described in Sections 3.4.1 and 3.4.2, respectively.

3.5. Discussion

The computational complexity of DyAdvDefender depends on the features used and the number of samples indexed. It is proportional to the sizes of the features because the feature of a query sample is compared with those of the indexed samples. The computational complexity of DyAdvDefender also depends on the feature types (e.g., integer and float) because their computational costs are different. As we stated above, the average number of indexed samples to compare with a query sample is $2^{-k}n$. If the indexed samples are distributed on z computing units, the computation of distances between the indexed samples and a query could be done by the z computing units simultaneously, with each comparing

Algorithm 3: Optimized DyAdvDefender.

```
Data: a query $\boldsymbol{x}$, the machine learning model to defend $C$, a set of indexed samples $S$, a distance threshold $\theta_0^*$
Result: Classification result $\boldsymbol{y}$
$S_0 = Potential\_Same\_Origin(S, \boldsymbol{x})$;
$min\_sam = None$;
$output = None$;
$min\_dist = \infty$;
for $(\boldsymbol{x_0}, \boldsymbol{y_0}) \in S_0$ do
    if $D(\boldsymbol{x_0}, \boldsymbol{x}) < min\_dist$ then
        $min\_dist = D(\boldsymbol{x_0}, \boldsymbol{x})$;
        $min\_sam = \boldsymbol{x_0}$;
        $output = \boldsymbol{y_0}$;
    end
end
if $min\_dist < \theta_0^*$ then
    $\boldsymbol{y} = output$;
    if $min\_dist \geq \theta_0^*/10$ then
        $Index((\boldsymbol{x}, \boldsymbol{y}), S)$;
    end
    Return $\boldsymbol{y}$;
else
    $\boldsymbol{y} = C(\boldsymbol{x})$;
    $Index((\boldsymbol{x}, \boldsymbol{y}), S)$;
    Return $\boldsymbol{y}$;
end
```

the query sample with $2^{-k}n/z$ different indexed samples. It means that z computing units could give approximately z times speedup compared with a single node. Thus, DyAdvDefender is linearly scalable.

In addition to the optimizations we previously provided, there could be others in real-world application scenarios. For example, the indexing database of DyAdvDefender could be cleared once a day or whenever the number of indexed samples reaches a certain value. That would barely impair the effectiveness of DyAdvDefender because the adversary could only get information at the first query after each database clearing, and a successful attack requires at least hundreds or thousands of informative queries.

4. Experiments

We conduct experiments to evaluate DyAdvDefender. Specifically, we try to answer the following research questions:

- ***RQ1***. How effective is DyAdvDefender in defending adversarial attacks compared with previous state-of-the-art defenses?
- ***RQ2***. Does DyAdvDefender affect the classification accuracy on natural samples?
- ***RQ3***. How does the average response time grow with the number of indexed samples?
- ***RQ4***. How does the number of LSH functions k affect the classification accuracy, attack success rate, and average response time?
- ***RQ5***. How does the threshold θ_0 used in the algorithm of DyAdvDefender affect the classification accuracy and attack success rate?

Our experiments are conducted on a server with one Intel(R) Core(TM) i9-9980XE CPU, 128 GB memory, and an Nvidia GeForce RTX 2080 Ti Graphics Card.

4.1. Datasets

We conduct experiments on two different domains of datasets: image classification and malware detection. They are representative fields with different characteristics in terms of the constraints on perturbation. In image classification, the samples are images, and the perturbation is performed on the values of pixels. The only constraint is that the perturbed values stay in the legitimate range. In malware detection, the samples are executables. There are different choices for the feature sets, and they cannot be perturbed arbitrarily because the file format should be kept, and the logic should be intact.

The most commonly used image data sets for adversarial attack and defense are MNIST and CIFAR-10, and we follow this convention. Both datasets have 10 classes of samples, and the greyscale or RBG values are standardized to the range [-0.5,0.5]. The two datasets have 10,000 samples in the test set. We reserve 500 samples from the test sets for adversarial attacks and the remaining 9,500 samples for evaluating the classification accuracy on the natural samples. The adversarial attacks are conducted on the reserved samples that are correctly classified before the attacks [19, 16, 26, 28, 40]. We collected a malware detection dataset that consists of 5,280 benign executables and 5,280 malicious executables. It has 8,448 samples in the training set, 1,056 in the validation set, and 1,056 in the test set. Following the literature [3, 12], the objective of the attack is always to trick the target system into classifying malicious executables as benign. We consider three different feature sets: PE header numerical fields, frequencies of printable strings, and imports of DLL functions [5, 24]. PE header numerical field features are standardized to the range [-0.5,0.5] because each is at a different scale. We reserve 200 malware samples from the test set for adversarial attacks. The adversarial attacks are conducted on the reserved samples that are correctly classified if not attacked.

4.2. Evaluation Metrics

Two main evaluation metrics are the *adversarial attack success rate* (*ASR*) and the *classification accuracy* (*ACC*) on natural samples. Following the literature, the attack success rate is the percentage of samples that are correctly classified before the perturbations by an adversarial algorithm but are misclassified after all the perturbations have been performed by an attacker. This metric answers ***RQ1*** and is our main focus because the primary objective of a defense mechanism is to reduce the attack success rate. The other metric is the classification accuracy on natural samples. This metric answers ***RQ2***. We compare the accuracy of the system defended by DyAdvDefender and the undefended system to see whether DyAdvDefender affects the classification accuracy. For other defense methods in our experiments that use different neural networks and may also be pretrained differently, the classification accuracy comparison between those systems and the system defended by DyAdvDefender is irrelevant to the scope of this study.

4.3. Two-Round Evaluation

In previous research [19, 16, 26, 28], the evaluation was first performed as a "one-time attack and defense game", then the attack success rate or the classification accuracy on the adversarial samples was computed. This setting is sufficient for evaluating previous defense methods because they are static and do not change their states as more query samples arrive. To properly evaluate DyAdvDefender as an instance-based online machine learning model, we design a two-round evaluation procedure.

In the first round, we query the classification system on natural samples to evaluate its classification accuracy. Then, we clear the indexed samples of DyAdvDefender and use an adversarial algorithm to attack the system with the reserved samples for adversarial attacks to evaluate the attack success rate. The classification accuracy of the defended system is compared with the undefended one, and the difference represents whether the accuracy is affected by our defense mechanism DyAdvDefender. The attack success rate of the defended system is also compared with the undefended one, and the difference represents the effectiveness of DyAdvDefender.

The objective of the second round is to observe (i) how adversarial samples affect the defended system in terms of classification accuracy on natural samples and (ii) how natural samples affect the defense effectiveness with adversarial attacks. Therefore, in the second round, DyAdvDefender keeps the samples indexed during the adversarial attacks in the first round, and then we query the system with natural samples to evaluate its classification accuracy. This achieves objective (i). We then clear the indexed samples and query the system with the natural samples again to let DyAdvDefender index them, and then we use an adversarial algorithm to attack the system with the reserved samples to evaluate the attack success rate. This achieves objective (ii).

It should be noted that when we evaluate the undefended system or other defense methods, we also follow a two-round evaluation. The difference is that there is no step for clearing the indexed samples for them, because they do not index anything and the states of those systems do not change in the two rounds. Hence, the classification accuracy and attack success rate in the two rounds should be approximately the same.

4.4. Adversarial Attack and Defense Methods

We use the following perturbation trial-based black-box adversarial attack methods to demonstrate the effectiveness of DyAdvDefender: ZOO, ZOO_AE, AutoMZOOM, and BLZOOM with their default settings to attack a deep learning model for image classification tasks [10, 40].

- ***ZOO*** is a coordinate-wise gradient estimation-based black-box attack method [10]. It perturbs one variable each time and accumulates the perturbations until a successful adversarial sample is found or a preset number of iterations has been tried.
- ***ZOO_AE*** is an improved version of ZOO [40]. It uses an autoencoder to learn a compressed representation of a sample. The perturbation is performed on the compressed representation.
- ***AutoMZOOM*** is a more significantly improved version of ZOO [40]. In addition to using an autoencoder, AutoZOOM adopts an adaptive random gradient estimation strategy to improve the efficiency.
- ***BLZOOM*** is a variant of AutoZOOM [40]. It uses a bilinear resizing operation to replace the autoencoder in AutoZOOM.

We set 1.0E-2 as the per-pixel perturbation bound on images to keep their original semantics. We discuss the validity of this setting in Section 4.11.

The integrity of the PE format and the original functionalities need to be kept when perturbing malware samples, and therefore there are constraints on the perturbations [3]. A perturbation on a compressed representation corresponds to the changes of multiple features, so the constraints cannot be guaranteed. Therefore, we can only use ZOO to attack a deep learning classifier for malware detection while following the constraints. As the constraints have already guaranteed that the original functionalities are kept, we do not set a perturbation bound for malware samples.

We compare DyAdvDefender with the following state-of-the-art defense methods [19, 1, 26, 28]:

- ***Grosse et al.*** [19] augment a classification model with an additional class for adversarial samples and train the model to classify adversarial examples to that class. We implement their approach and apply it in the experiments with all datasets.
- ***Akhtar et al.*** [1] learns a Perturbation Rectifying Network (PRN) to reconstruct an image from a potentially perturbed one. Then, a perturbation detector is trained on the Discrete Cosine Transform of the input-output difference of the PRN. If the detector determines that an image is perturbed, the reconstructed image is fed to the original classifier; otherwise, the original image is fed.
- ***Madry et al.*** [26] prove that PGD is the strongest first order white-box attack and propose an objective function to train a network to be robust against PDG attacks so that it can also defend against other inferior attacks. The authors provide the pre-trained robust models for the MNIST and CIFAR-10 datasets, which are different from the models defended by DyAdvDefender, so we include the comparison on the reduction of the attack success rate with their methods on those two datasets.
- ***Musta et al.*** [28] propose a new objective function called Prototype Conformity Loss (PCL) to train a neural network. It forces the features of samples in different classes to have a large distance with each other, which makes the adversarial perturbations more difficult to fulfill their objectives. The authors provide the source code only for the experiments with CIFAR-10 and use a different neural network; therefore, we compare the reduction of the attack success rate with their defense only on CIFAR-10.

4.5. Classification Models to Defend

For the undefended classification system, the classification system defended by DyAdvDefender, and the system defended by the method proposed by Grosse et al. [19], the target models on the two image datasets are the two convolutional neural networks used in [40]. We directly download their pretrained models for our experiments.

Table 1
The architectures of the neural networks defended by DyAdvDefender for malware detection.

Feature	Header	Strings	Imports
Input	54	4,651	10,201
Layer 1	$Relu(FC(32))$	$Relu(FC(64))$	$Relu(FC(128))$
Layer 2	$Relu(FC(16))$	$Relu(FC(32))$	$Relu(FC(64))$
Layer 3	$Softmax(FC(2))$	$Softmax(FC(2))$	$Softmax(FC(2))$

Table 2
The distance threshold θ_0^* computed on each dataset and each feature set.

Dataset (Feature)	Distance measure	θ_0^*
MNIST	Euclidean	0.82
CIFAR-10	Euclidean	2.75
Malware Detection (header)	Euclidean (Standardized)	0.28
Malware Detection (imports)	Manhattan	2
Malware Detection (strings)	Manhattan	8

For the other defense methods that we compare with [26, 28], the networks are given in their papers and source code. We use the models that they provide to conduct the experiments. Because they use different neural networks, the comparison with those models on classification accuracy on natural samples is not relevant to the performance of defense mechanisms, and we make our comparison with them only on the reduction of the attack success rate.

The undefended models for malware detection are feed-forward neural networks (FNNs) with two hidden layers applied on those features respectively. They are trained with cross entropy as the objective function and Adam as the optimization algorithm. The details of the architectures are presented in Table 1.

4.6. Hyperparameters

The distance threshold θ_0^* values computed on the training sets are shown in Table 2.

We set the number of LSH functions $k = 10$ in our basic experiments, which is sufficient for even larger datasets than the ones we used in the experiments. We also show how the changes of k affect the performance in all respects in Section 4.8.

4.7. Results

As the adversarial algorithms have randomness, we run each experiment three times and report the mean. The classification accuracy on natural samples and attack success rate in the two rounds are presented in Table 3 for MNIST, Table 4 for CIFAR-10, and Table 5 for malware detection.

As shown, all defense methods could reduce the ASR of all black-box attack methods to different degrees in the experiments. Overall, our defense method DyAdvDefender significantly outperforms all other defense mechanisms in reducing the ASR, especially for MNIST, where the ASR is reduced to almost 0%. Akhtar et al. [1] is the runner-up defense method. It has the same effectiveness as DyAdvDefender in reducing the ASR to defend against ZOO_AE and BLZOOM on MNIST, and it is only inferior in some other settings. However, it is much less effective in defending against three kinds of attacks (i.e., ZOO, ZOO_AE, AUTOZOOM) on CIFAR-10. The other defense methods have a larger gap with DyAdvDefender in the defense effectiveness. The unique characteristic of DyAdvDefender as an online machine learning and instanced-based model allows it to dynamically adapt itself to the coming attacks, and thus it outperforms static methods. This result provides an answer to ***RQ1***.

On natural samples, the ACC of a classification system defended by DyAdvDefender is the same as the undefended one. This suggests that DyAdvDefender does not affect the classification accuracy on natural samples and provides an answer to ***RQ2***. The defense method proposed by Akhtar et al. [1] is applied to the same classification model, and we can see that their method does not affect the ACC either. It should be noted that the ACC for the other defense methods

Table 3
Experiment results on MNIST.

Attack	Defense	Round 1		Round 2	
		ACC	ASR	ACC	ASR
ZOO	No defense	99.4%	99.0%	99.4%	99.0%
	DyAdvDefender	99.4%	**0%**	99.4%	**0%**
	Akhtar et al.	99.4%	9.1%	99.4%	8.9 %
	Grosse et al.	99.2%	77.6%	99.2%	77.3%
	Madry et al.	98.4%	9.2%	98.4%	9.5%
ZOO_AE	No defense	99.4%	99.2%	99.4%	99.2%
	DyAdvDefender	99.4%	**0%**	99.4%	**0%**
	Akhtar et al.	99.4%	**0%**	99.4%	**0%**
	Grosse et al.	99.2%	81.3%	99.2%	81.7%
	Madry et al.	98.4%	99.3%	98.4%	99.4%
BLZOOM	No defense	99.4%	99.2%	99.4%	99.4%
	DyAdvDefender	99.4%	**0%**	99.4%	**0.1%**
	Akhtar et al.	99.4%	1.6%	99.4%	1.7%
	Grosse et al.	99.2%	51.6%	99.2%	51.3%
	Madry et al.	98.4%	64.4%	98.4%	64.6%
AUTOZOOM	No defense	99.4%	99.6%	99.4%	99.4%
	DyAdvDefender	99.4%	**0.2%**	99.4%	**0%**
	Akhtar et al.	99.4%	2.2%	99.4%	2.2%
	Grosse et al.	99.2%	68.1%	99.2%	68.3%
	Madry et al.	98.4%	80.4%	98.4%	80.2%

do not change in the two rounds because the classification model to protect is a deterministic algorithm, and the ASR may change because the adversarial algorithms are randomized.

One observation is that the effectiveness of the defense methods depends on the dataset and adversarial algorithm. CIFAR-10 is harder to defend than MNIST for all defenses. Yet, there is no such consistency in adversarial algorithms. Compared with ZOO and ZOO_AE, AutoZOOM and BLZOOM are harder for DyAdvDefender to defend against but easier for the defenses proposed by Grosse et al. [19] and Mustafa et al. [28]. Madry et al. [26] defend against ZOO best and ZOO_AE the worst. Akhtar et al. [1] defend against all attacks well on MNIST and defend against AutoZOOM and BLZOOM well on CIFAR-10.

On malware detection, the defense effectiveness of DyAdvDefender and the defense proposed by Grosse et al. [19] varies from one feature set to another, e.g., they both defend better with printable strings as the features. This confirms our previous statement that the choice of the feature set could make a significant difference.

4.8. Efficiency Study

The overhead that DyAdvDefender brings to the classification system mainly comes from the procedure to find the indexed samples that potentially have the same origin as a query sample. Therefore, theoretically the average response time (ART) grows with the number of samples indexed in the system. That is why we use the LSH-based algorithm to optimize the efficiency. We used all the samples of MNIST and CIFAR-10 to evaluate DyAdvDefender with respect to its efficiency. Figure 1 depicts the relation between the average response time and the number of indexed samples. As expected, from Figure 1 1.a) and 2.a), we can see that the average response time of DyAdvDefender without LSH grows linearly with the number of indexed samples (i.e., $ART_{NoLSH}(N) = k_1 N + b_1$). This is because we need to compute the distance between all indexed samples and a query sample. In comparison, the growth of the average response time of the optimized DyAdvDefender with LSH can be ignored. To see it more clearly, we show the growth of the logarithm of the average response time for DyAdvDefender with and without LSH, together with the average

Table 4
Experiment results on CIFAR-10.

Attack	Defense	Round 1		Round 2	
		ACC	ASR	ACC	ASR
ZOO	No defense	77.9%	98.5%	77.9%	98.5%
	DyAdvDefender	77.9%	**0%**	77.9%	**0%**
	Akhtar et al.	77.9%	81.1%	77.9%	81.0%
	Grosse et al.	72.8%	98.2%	72.8%	98.2%
	Madry et al.	87.3%	71.4%	87.3%	71.5%
	Musta et al.	89.0%	75.5%	89.0%	75.5%
ZOO_AE	No defense	77.9%	98.8%	77.9%	98.8%
	DyAdvDefender	77.9%	**1.3%**	77.9%	**1.2%**
	Akhtar et al.	77.9%	79.6%	77.9%	79.9%
	Grosse et al.	72.8%	97.9%	72.8%	98.0%
	Madry et al.	87.3%	98.6%	87.3%	98.4%
	Musta et al.	89.0%	52.3%	89.0%	52.5%
BLZOOM	No defense	77.9%	99.5%	77.9%	99.3%
	DyAdvDefender	77.9%	**2.7%**	77.9%	**2.9%**
	Akhtar et al.	77.9%	14.6%	77.9%	14.6 %
	Grosse et al.	72.8%	95.2%	72.8%	94.8%
	Madry et al.	87.3%	86.5%	87.3%	86.3%
	Musta et al.	89.0%	45.1%	89.0%	45.5%
AUTOZOOM	No defense	77.9%	99.8%	77.9%	99.8%
	DyAdvDefender	77.9%	**14.2%**	77.9%	**14.2%**
	Akhtar et al.	77.9%	39.7%	77.9%	39.5%
	Grosse et al.	72.8%	97.1%	72.8%	97.2%
	Madry et al.	87.3%	94.6%	87.3%	94.6%
	Musta et al.	89.0%	37.5%	89.0%	36.8%

Table 5
Experimental results on malware detection. The attack method is ZOO.

Feature	Defense	Round 1		Round 2	
		ACC	ASR	AC	ASR
Header	No defense	96.8%	100%	96.8%	100%
	DyAdvDefender	96.8%	**73.5%**	96.8%	**73.5%**
	Grosse et al.	96.2%	99.0%	96.2%	99.0%
Imports	No defense	98.8%	100%	98.8%	100%
	DyAdvDefender	98.8%	**40.0%**	98.8%	**39.3%**
	Grosse et al.	98.5%	99.8%	98.5%	99.7%
Strings	No defense	98.5%	100%	98.5%	100%
	DyAdvDefender	98.5%	**6.0%**	98.5%	**5.3%**
	Grosse et al.	98.2%	71.2%	98.2%	70.8%

response time of the classification system without defense in Figure 1 1.b) and 2.b). We can see that the logarithm of the average response time of DyAdvDefender with LSH stays at the same magnitude as the number of indexed samples grows. This means that the overhead brought by DyAdvDefender is well bounded as the number of indexed samples grows. Even though the overhead is still greater than the parametric defense methods as opposed to instance-based methods, the advantage of our defense effectiveness is worth the bounded overload. Thus, it is ideally practical for real-world applications, and as a result we have answered ***RQ3***.

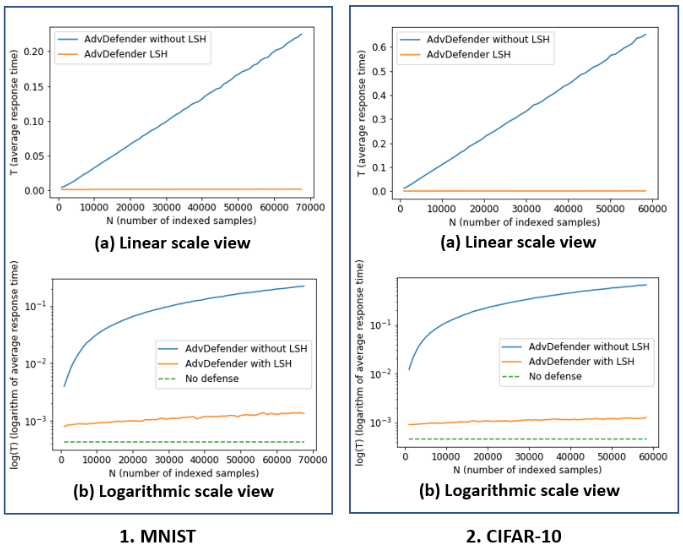


Figure 1: The relation between the average response time and the number of indexed samples on MNIST and CIFAR-10. The average response time is shown on both linear scale view and logarithmic scale view.

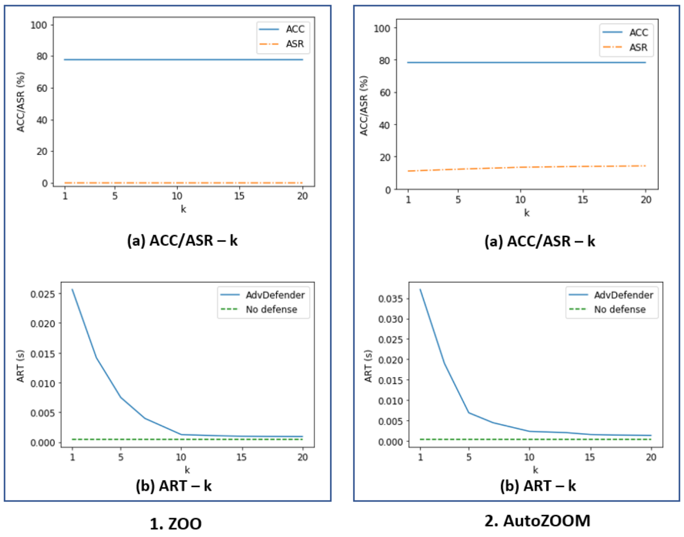
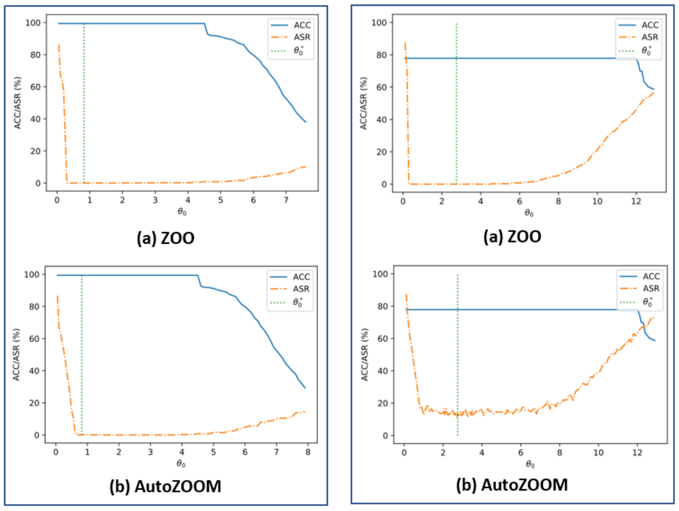


Figure 2: The relation between the classification accuracy, attack success rate, and average response time with the number of LSH functions, with ZOO and AutoZOOM as the attacks.

4.9. Impacts of Number of LSH Functions

In Figure 2, we present the impacts of the number of LSH functions k to the average response time, classification accuracy, and attack success rate of the classification system defended by DyAdvDefender. We use ZOO and AutoZOOM as the adversarial algorithms on CIFAR-10 in these experiments, because they are the best and worst defended adversarial algorithms by DyAdvDefender. We observe that within the range we test, from 1 LSH function to 20 LSH functions, the average response time drops significantly before it reaches 10 LSH functions. In contrast, the classification accuracy on natural samples is not affected at all. As the number of LSH functions increases, the attack success rate is not affected with ZOO and grows very slowly with AutoZOOM. The defense effectiveness is still excellent even when the number of LSH functions is 20, which keeps the average number of indexed samples to compare with a query sample at 1,024 when 1 billion samples are indexed. Therefore, depending on the number of indexed samples in real-world scenarios, classification service providers can improve efficiency by increasing the number of LSH functions



1. MNIST **2. CIFAR-10**

Figure 3: The relation between ASR/ACC and θ_0 on MNIST and CIFAR-10. The empirical distance threshold θ_0^* computed on the training sets is shown as vertical lines.

without worrying about a significant drop of defense effectiveness within a large range. This provides an answer to ***RQ4***.

4.10. Impacts of Threshold θ_0

Figure 3 depicts the impacts of the distance threshold θ_0 that is set in DyAdvDefender to ASR and ACC. We can see that there is a range of θ_0 that keeps DyAdvDefender at its optimal performance (i.e., without loss of ACC and at its lowest ASR). Such an optimal range exists because samples that have small distances are from the same classes, and even if the defense mechanism determines them as from the same origin, the classification result would not be affected. As can be seen, the empirical θ_0^* computed on the training sets is always within the optimal range. Figure 4 shows the number of sample pairs of which the distances are within the optimal ranges that are shown in Figure 3. In both datasets, there are at least tens of thousands of sample pairs of which the distances are within the optimal range of θ_0. Therefore, the proposed defense method is robust so that even if some samples are removed from the training set, DyAdvDefender could still keep its optimal performance.

As θ_0 decreases beyond the optimal range, the ASR increases, which means DyAdvDefender fails to defend against the attacks more frequently. This is because when θ_0 is set to be smaller than the perturbation, DyAdvDefender would consider that the perturbed sample and the original sample have different origins, thus the attack bypasses the defense mechanism. Yet, this does not affect the classification on original samples. As θ_0 increases beyond the optimal range, we can see the ASR increases and the ACC decreases. This is because as the θ_0 gets larger, samples in different classes are considered having the same origin by DyAdvDefender, so it outputs wrong classification results which negatively impacts both ASR and ACC. This result answers ***RQ5***.

4.11. Validity of Adversarial Samples

We ensure that the adversarial malware samples keep their original semantics by following the rules of perturbation. However, the adversarial samples of images cannot be verified with rules. Therefore, we manually examine the adversarial samples crafted by attacking the defended methods. We can still correctly recognize all their original classes, even though for some of them the perturbations are clearly perceptible. Figure 5 shows some examples of successful attacks with the per-pixel perturbation between 1.0E-3 and 1.0E-2. We can see that the adversarial samples of MNIST contain some noise points, and those of CIFAR-10 seem to contain a "melted" area, but they still keep their original semantics. However, further perturbations may begin to change the semantics of the adversarial samples. This result suggests that our setting of the perturbation threshold is valid.

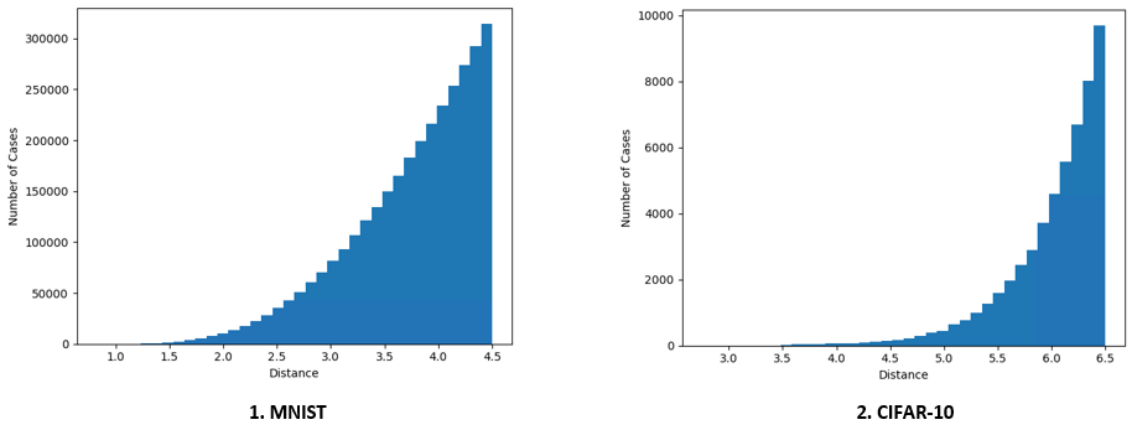


Figure 4: The histograms show the numbers of sample pairs whose distances are within the optimal θ_0 range.

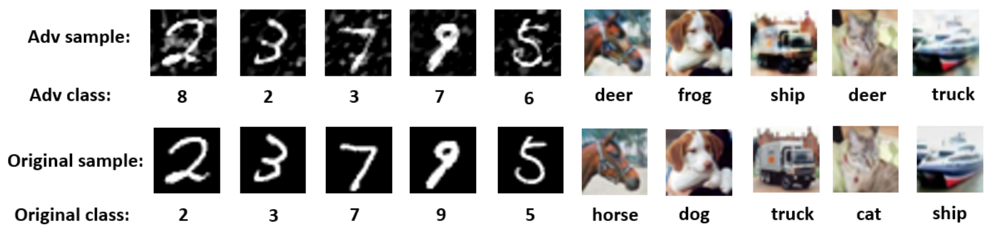


Figure 5: Adversarial samples with per-pixel perturbation between 1.0E-3 and 1.0E-2 and their original samples.

5. Limitations

The value of θ_0^* is computed based on a training set. As demonstrated in our experiments, θ_0^* that is computed with our approach is within the range that corresponds to the best defense performance. This is because the datasets for research use are usually large enough and clean. They are the requirements for the proposed defense method to work at its best performance. However, in real-world applications, a training set may not be perfectly curated, e.g., adversarial samples and wrong labelled samples may exist, and the training set may not be large enough. Then, the θ_0^* may go out of the most effective range. Therefore, in real-world applications a manual examination of the samples with the minimal distances and adjustment on θ_0^* may be needed so that DyAdvDefender can achieve its optimal performance.

The proposed defense mechanism DyAdvDefender relies on the comparison of feature vectors of a query sample and the indexed samples. A bad choice of the feature set may cause the defense mechanism to fail. As an extreme example, with a bad choice of the feature set, two samples from different classes could have the exact feature vectors. The machine learning classification model would not even be able to differentiate them, and the DyAdvDefender would fail too. If the machine learning model to defend is based on multiple feature sets, it is better to deploy DyAdvDefender on each feature set separately, rather than uniting all feature sets as one set, because the former could remove the effects of the difference on the distance scale across different feature sets.

6. Conclusion

In this paper, we present a novel state-of-the-art black-box adversarial attack defense method called DyAdvDefender. As an online machine learning and instance-based defense method, it updates its states to adapt to attacks it has received. Experimental results suggest that DyAdvDefender outperforms previous state-of-the-art methods in defending against perturbation trial-based black-box adversarial attacks, without harming the classification accuracy on natural samples. To optimize its efficiency, we propose an LSH-based solution for indexing and retrieving samples. Experimental results also illustrate that DyAdvDefender is efficient for practical use. As the attacks on substitute models have already been defended efficiently, our approach serves as an excellent complement for the overall black-box adversarial attack defense problem.

Acknowledgment

This research is supported by Defence Research and Development Canada (contract no. W7701-176483), the Discovery Grants (RGPIN-2018-03872) from the Natural Sciences and Engineering Research Council of Canada, and Canada Research Chairs Program (950-232791).

References

[1] Akhtar, N., Liu, J., Mian, A., 2018. Defense against universal adversarial perturbations, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3389–3398.

[2] Alexandr, A., Piotr, I., 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Communications of The ACM .

[3] Anderson, H.S., Kharkar, A., Filar, B., Roth, P., 2017. Evading machine learning malware detection. Black Hat .

[4] Athalye, A., Carlini, N., Wagner, D., 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. arXiv preprint arXiv:1802.00420 .

[5] Baldangombo, U., Jambaljav, N., Horng, S.J., 2013. A static malware detection system using data mining methods. arXiv preprint arXiv:1308.2831 .

[6] Bawa, M., Condie, T., Ganesan, P., 2005. Lsh forest: self-tuning indexes for similarity search, in: Proceedings of the 14th International Conference on World Wide Web, ACM. pp. 651–660.

[7] Byerly, A., Kalganova, T., Dear, I., 2020. A branching and merging convolutional network with homogeneous filter capsules. arXiv preprint arXiv:2001.09136 .

[8] Carlini, N., Wagner, D., 2017. Towards evaluating the robustness of neural networks, in: Proceedings of the IEEE Symposium on Security and Privacy (S&P), IEEE. pp. 39–57.

[9] Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., Mukhopadhyay, D., 2018. Adversarial attacks and defences: A survey. arXiv preprint arXiv:1810.00069 .

[10] Chen, P.Y., Zhang, H., Sharma, Y., Yi, J., Hsieh, C.J., 2017. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models, in: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, ACM. pp. 15–26.

[11] Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., Usunier, N., 2017. Parseval networks: Improving robustness to adversarial examples, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org. pp. 854–863.

[12] Demetrio, L., Biggio, B., Lagorio, G., Roli, F., Armando, A., 2019. Explaining vulnerabilities of deep learning to adversarial malware binaries. arXiv preprint arXiv:1901.03583 .

[13] Dhillon, G.S., Azizzadenesheli, K., Lipton, Z.C., Bernstein, J., Kossaifi, J., Khanna, A., Anandkumar, A., 2018. Stochastic activation pruning for robust adversarial defense. arXiv preprint arXiv:1803.01442 .

[14] Ding, S.H.H., Fung, B.C.M., Charland, P., 2016. Kam1n0: Mapreduce-based assembly clone search for reverse engineering, in: Proceedings of the 22nd ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), pp. 461–470.

[15] Fang, H., Zhu, G., Stojanovic, V., Nie, R., He, S., Luan, X., Liu, F., 2021. Adaptive optimization algorithm for nonlinear markov jump systems with partial unknown dynamics. International Journal of Robust and Nonlinear Control 31, 2126–2140.

[16] Feinman, R., Curtin, R.R., Shintre, S., Gardner, A.B., 2017. Detecting adversarial samples from artifacts. arXiv preprint arXiv:1703.00410 .

[17] Goemans, M.X., Williamson, D.P., 1995. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of the ACM (JACM) 42, 1115–1145.

[18] Goodfellow, I.J., Shlens, J., Szegedy, C., 2014. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 .

[19] Grosse, K., Manoharan, P., Papernot, N., Backes, M., McDaniel, P., 2017. On the (statistical) detection of adversarial examples. arXiv preprint arXiv:1702.06280 .

[20] Gu, S., Rigazio, L., 2014. Towards deep neural network architectures robust to adversarial examples. arXiv preprint arXiv:1412.5068 .

[21] Ilyas, A., Engstrom, L., Athalye, A., Lin, J., 2018. Black-box adversarial attacks with limited queries and information. arXiv preprint arXiv:1804.08598 .

[22] Indyk, P., Motwani, R., 1998. Approximate nearest neighbors: towards removing the curse of dimensionality, in: Proceedings of the 30th annual ACM Symposium on Theory of Computing, ACM. pp. 604–613.

[23] Kurakin, A., Goodfellow, I., Bengio, S., 2016. Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533 .

[24] Li, M.Q., Fung, B.C.M., Charland, P., Ding, S.H.H., 2021. I-mad: Interpretable malware detector using galaxy transformer. Computers & Security , 102371.

[25] Lu, J., Issaranon, T., Forsyth, D., 2017. Safetynet: Detecting and rejecting adversarial examples robustly, in: Proceedings of the IEEE International Conference on Computer Vision, pp. 446–454.

[26] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A., 2019. Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083v4 .

[27] Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P., 2016. Deepfool: a simple and accurate method to fool deep neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2574–2582.

[28] Mustafa, A., Khan, S., Hayat, M., Goecke, R., Shen, J., Shao, L., 2019. Adversarial defense by restricting the hidden space of deep neural networks, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 3385–3394.

[29] Narodytska, N., Kasiviswanathan, S.P., 2016. Simple black-box adversarial perturbations for deep networks. arXiv preprint arXiv:1612.06299 .

[30] Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A., 2017. Practical black-box attacks against machine learning, in: Proceedings of the 2017 ACM Asia Conference on Computer and Communications Security (ASIA CCS), pp. 506–519.

[31] Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A., 2016a. The limitations of deep learning in adversarial settings, in: Proceedings of the 2016 IEEE European Symposium on Security and Privacy (EuroS&P), IEEE. pp. 372–387.

[32] Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A., 2016b. Distillation as a defense to adversarial perturbations against deep neural networks, in: Proceedings of the 2016 IEEE Symposium on Security and Privacy (S&P), IEEE. pp. 582–597.

[33] Pereira, G.T., de Carvalho, A.C., 2019. Bringing robustness against adversarial attacks. Nature Machine Intelligence 1, 499–500.

[34] Shaham, U., Yamada, Y., Negahban, S., 2018. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. Neurocomputing 307, 195–204.

[35] Stojanovic, V., He, S., Zhang, B., 2020. State and parameter joint estimation of linear stochastic systems in presence of faults and non-gaussian noises. International Journal of Robust and Nonlinear Control 30, 6683–6700.

[36] Su, J., Vargas, D.V., Sakurai, K., 2019. One pixel attack for fooling deep neural networks. IEEE Transactions on Evolutionary Computation 23, 828–841.

[37] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R., 2013. Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 .

[38] Tang, J., Du, X., He, X., Yuan, F., Tian, Q., Chua, T.S., 2019. Adversarial training towards robust multimedia recommender system. IEEE Transactions on Knowledge and Data Engineering 32, 855–867.

[39] Tao, H., Li, J., Chen, Y., Stojanovic, V., Yang, H., 2020. Robust point-to-point iterative learning control with trial-varying initial conditions. IET Control Theory & Applications 14, 3344–3350.

[40] Tu, C.C., Ting, P., Chen, P.Y., Liu, S., Zhang, H., Yi, J., Hsieh, C.J., Cheng, S.M., 2019. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 742–749.

[41] Wang, D., Li, C., Wen, S., Nepal, S., Xiang, Y., 2020. Defending against adversarial attack towards deep neural networks via collaborative multi-task training. IEEE Transactions on Dependable and Secure Computing .

[42] Wolfe, P., 1982. Checking the calculation of gradients. ACM Transactions on Mathematical Software (TOMS) 8, 337–343.

[43] Xie, C., Wu, Y., Maaten, L.v.d., Yuille, A.L., He, K., 2019. Feature denoising for improving adversarial robustness, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 501–509.

[44] Xu, H., Ma, Y., Liu, H., Deb, D., Liu, H., Tang, J., Jain, A., 2019. Adversarial attacks and defenses in images, graphs and text: A review. arXiv preprint arXiv:1909.08072 .

[45] Xu, W., Evans, D., Qi, Y., 2017. Feature squeezing: Detecting adversarial examples in deep neural networks. arXiv preprint arXiv:1704.01155 .

[46] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R.R., Le, Q.V., 2019. Xlnet: Generalized autoregressive pretraining for language understanding, in: Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), pp. 5754–5764.

[47] Yu, P., Song, K., Lu, J., 2018. Generating adversarial examples with conditional generative adversarial net, in: Proceedings of 24th International Conference on Pattern Recognition (ICPR), IEEE. pp. 676–681.

[48] Zhou, Y., Kantarcioglu, M., Thuraisingham, B., Xi, B., 2012. Adversarial support vector machine learning, in: Proceedings of the 18th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), pp. 1059–1067.

[49] Zhu, D., Zhang, Z., Cui, P., Zhu, W., 2019. Robust graph convolutional networks against adversarial attacks, in: Proceedings of the 25th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), pp. 1399–1407.

[50] Zügner, D., Borchert, O., Akbarnejad, A., Guennemann, S., 2020. Adversarial attacks on graph neural networks: Perturbations and their patterns. ACM Transactions on Knowledge Discovery from Data (TKDD) 14, 1–31.

Miles Q. Li is a Ph.D. candidate in the School of Computer Science, McGill University, Montreal, Canada. He received his B.Sc and M.Sc from Peking University. His research interests include data mining for cybersecurity, deep learning, and natural language processing.

Benjamin C. M. Fung received his Ph.D. degree in Computing Science from Simon Fraser University in Canada in 2007. He is a Canada Research Chair in Data Mining for Cybersecurity and a Professor with the School of Information Studies at McGill University in Canada. He also serves as Associate Editor for IEEE Transactions of Knowledge and Data Engineering (TKDE) and Elsevier Sustainable Cities and Society (SCS). He has over 130 refereed publications, with more than 11,000 citations, that span the research forums of data mining, privacy protection, cybersecurity, services computing, and building engineering. Prof. Fung is also a licensed Professional Engineer of software engineering in Ontario, Canada.

Philippe Charland is a Defence Scientist at Defence Research and Development Canada – Valcartier Research Centre, in the Mission Critical Cyber Security Section. His research interests encompass software reverse engineering and computer forensics. As a member of the Systems Vulnerabilities and Lethality Group, his research focuses on binary-level program comprehension to accelerate the reverse engineering process involved in malicious software analysis and classification, as well as for mission assurance. Mr. Charland holds a bachelor and a master's degree in Computer Science, both from Concordia University, Montreal, Canada.