K-Anonymity inspired Adversarial Attack and Multiple One-Class Classification Defense

Vasileios Mygdalis, Anastasios Tefas, Ioannis Pitas

Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece
{pitas, tefas} csd.auth.gr
Author pre-print version, submitted to Elsevier Neural Networks.
Please cite the publisher maintained version in your work.
DOI : https://doi.org/10.1016/j.neunet.2020.01.015

Abstract

A novel adversarial attack methodology for fooling deep neural network classifiers in image classification tasks is proposed, along with a novel defense mechanism to counter such attacks. Two concepts are introduced, namely the K-Anonymity-inspired Adversarial Attack (K-A^3) and the Multiple Support Vector Data Description Defense (M-SVDD-D). The proposed K-A^3 introduces novel optimization criteria to standard adversarial attack methodologies, inspired by the K-Anonymity principles. Its generated adversarial examples are not only misclassified by the neural network classifier, but are uniformly spread along K different ranked output positions. The proposed M-SVDD-D consists of a deep neural architecture layer consisting of multiple non-linear one-class classifiers based on Support Vector Data Description that can be used to replace the final linear classification layer of a deep neural architecture, and an additional class verification mechanism. Its application decreases the effectiveness of adversarial attacks, by increasing the noise energy required to deceive the protected model, attributed to the introduced non-linearity. In addition, M-SVDD-D can be used to prevent adversarial attacks in black-box attack settings.

Keywords: K-Anonymity, Adversarial Defense, Adversarial Attack, Deep SVDD, Kernel Learning

1. Introduction

In image classification tasks (e.g., face/object recognition), the term *adversarial examples* refers to crafted images that appear to the human eye almost imperceptibly similar to the training samples, while being misclassified by the respective image classifier. The attempt of crafting such examples to this end is the so-called *adversarial attack*. Many classification models, including the ones based on Convolutional Neural Networks (CNN), have been found to be vulnerable to adversarial attacks [1, 2, 3]. Furthermore, recent studies [4, 5, 6] have shown that adversarial attacks have the property of *transferability*, i.e., carefully crafted adversarial examples may deceive various classification methods at the same time, ranging from similar deep architectures to even totally different classification methods, such as Support Vector Machines or Random Forests.

The research community has been actively developing adversarial attack methodologies over the past few years, as well as methodologies to anticipate these attacks. Different types of adversarial attacks are

specified in the literature, depending on the level of information available to the adversary prior to the attack. In most cases, a white-box attack is assumed, i.e., the adversary has full knowledge about the model architecture to be deceived, including access to the values of the respective model weights. Therefore, the adversary is allowed to form queries to the model in order to backpropagate gradients for the given inputs by employing appropriate loss functions. In the black-box attack case, it is assumed that the adversary has limited or no information about the model architecture, other than its output classification labels. White-box attacks may be applied to attack model architectures unknown to the adversary, by employing intermediate/reference architectures (known to the adversary) and by exploiting the property of transferability [4]. The design of adversarial defense methods to repulse these attacks seems to be a lot more challenging than initially anticipated [7]. Recent adversarial defense methods were based on obfuscating the model gradients [8] for the given inputs, repulsing a number of known adversarial attack methodologies in white-box adversarial attack setups. However, it was later found that the defenses relying on obfuscating gradients are significantly less effective against newer and stronger attacks, or against transferability attacks generated by employing similar undefended architectures and can nowadays be easily overcome by the adversary [9].

In this paper, we consider that adversarial attacks should not only be viewed in a negative way as methods for fooling deep neural networks, as they have been used in other applications, notably in protecting private data automated analysis by recognition systems, that are typically used by service providers in social media [10]. For example, adversarial attacks have been employed to disable known automatic face detection/recognition algorithms applied on visual data uploaded by social media users [11], without severely compromising image quality [12], while at the same time, not hiding the person identities to human viewers. Moreover, adversarial attack methods could potentially be used to protect data captured from publicly installed cameras or even IoT sensors (e.g., UAV/surveillance/car cameras). However, to the best of our knowledge, unlike standard privacy protection methods [13], adversarial attack methodologies do not incorporate privacy protection-related constraints in their optimization process, therefore, even if they are successful in disabling face detection/recognition against a specific algorithm, there are no guarantees that adversarial attacks are effective for protecting people's privacy, when employed against automated classification systems to this end.

On the other hand, adversarial attacks could potentially be used for malicious purposes against classification systems in sensitive applications, e.g., biometrics, forensics, spam/fault detection systems, or even copyright protection systems. Classification systems that are not robust against adversarial attacks may be rendered unreliable for real-world deployment. In order to measure the potential threat, adversarial attacks could be employed by the classification system engineer as a measure to intuitively expose innate classification model weaknesses e.g., over-fitting, since their application reveals the decision noise tolerance, which is directly related to the amount of additive noise required to result in the misclassification decision. However, in order to be protected against adversaries, novel defense mechanisms should be employed in such classification systems that not only hinder adversarial example crafting, but also to increase the model's tolerance to noise and/or at least detect/prevent adversarial attacks as last resort option. To this end, we argue that one-class classification methods such as the Support Vector Data Description [14] can be used as an additional mechanism to verify if the input samples belongs to one of the training classes.

Motivated by the potential applications, we propose an extension of the use of adversarial attacks in order to fool deep neural network classifiers in a privacy preserving manner, along with a novel defense mechanism to counter them. Two concepts are introduced, namely the K-Anonymity-inspired Adversarial Attack (K-A^3) and the Multiple Support Vector Data Description [14] (M-SVDD) Defense. The novel contributions of this work can be summarized as follows:

- A novel adversarial attack optimization problem is proposed that exploits and extends well-known

adversarial attack methodologies, by modifying the optimization conditions for generating the adversarial examples. The proposed optimization problem is inspired by K-Anonymity principles, assuring that the initial identities of the crafted adversarial examples are not only misclassified by the neural network decision function, but are uniformly spread along K different ranked output positions.

- In order to minimize the introduced perturbation by our adversarial attack, a visual similarity loss is introduced, that guides the adversarial attack towards image pixel value modifications having minimal impact on the perceived image quality. The CW-SSIM loss [15] is employed to this end.

- A novel deep neural layer composed of a number of novel deep non-linear one-class classifiers (SVDD layer), equal to the number of classes supported by the model to be protected, is proposed as an adversarial defense mechanism. The parameters of the SVDD layer are trained by exploiting novel loss functions inspired by the Support Vector Data Description [14]. The SVDD layer is thereby used to replace the standard linear classification layer of a pre-trained reference deep neural architecture, introducing non-linearity to the classifier decision function.

- In black-box attack settings, the proposed defense mechanism acts as input verification mechanism, i.e., ensures that if an input data vector does not belong to any of the training classes (i.e., is classified as outlier by every SVDD classifier), it is an adversarial example. The proposed defense mechanism can merely be a post-processing step after model inference, and does not hinder the application of other defense mechanisms at the same time.

2. Background information and related work

Let $\boldsymbol{x} \in \mathbb{R}^D$ be a general data sample (e.g., a facial image for face recognition) having a discrete ground truth label $y \in \mathcal{Y} = \{\ell_1, \ldots, \ell_C\}$ representing one of the C classes corresponding to e.g., facial image identities. Also let a neural network architecture consisting of L layers, having a trainable parameter set $\mathcal{W} = \{\boldsymbol{W}_i\}_{i=1}^{L}$, where $\boldsymbol{W}_i$ contains the i-th layer weights and also let a classifier decision function $f : \mathbb{R}^D \mapsto \mathbb{R}^C$ that maps the input samples to decision values, corresponding to each class. The sample $\boldsymbol{x}$ is classified correctly by the neural network classifier if $argmax(f(\boldsymbol{x}; \mathcal{W})) = y$.

2.1. Adversarial attacks

Let $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{n}$ denote an adversarial example that is crafted by perturbing $\boldsymbol{x}$. The general goal of adversarial attacks is to determine a noise vector $\boldsymbol{n}$ that is required to be added to $\boldsymbol{x}$, in order to change the classifier label, i.e.:

$$argmax(f(\tilde{\boldsymbol{x}}; \mathcal{W})) \neq y,$$

that is commonly determined by optimizing some objective function for the noise vector, e.g., minimizing its L_2 norm $\|\boldsymbol{n}\|_2$. For example, the L-BFGS attack [1] assumes access to the outputs of a continuous loss function denoted by $L_f : \mathbb{R}^C \times \mathcal{Y} \mapsto \mathbb{R}^+$, associated with the classifier function f to be deceived. The adversary selects a target label $t \neq y \in \mathcal{Y}$ for the adversarial example $\tilde{\boldsymbol{x}}$. Then, the following optimization problem is solved in iterative manner:

$$\min_{\boldsymbol{n}}: \quad c\|\boldsymbol{n}\|_2 + L_f(f(\tilde{\boldsymbol{x}}; \mathcal{W}), t), \tag{1}$$

until the minimum $\boldsymbol{n}$ that satisfies $argmax(f(\tilde{\boldsymbol{x}}; \mathcal{W})) = t$ is obtained (or approximated for non-convex loss functions L_f). The parameter $c > 0$ controls the amount of perturbation introduced per iteration step and is empirically set using line search [1]. Fast Gradient Sign [2] is a significantly faster alternative method that estimates $\boldsymbol{n}$ in a single optimization step along the direction of the gradient sign at each image pixel:

$$\boldsymbol{n} = c \cdot \text{sign}(\nabla L_f(f(\boldsymbol{x}; \mathcal{W}), t)), \tag{2}$$

at the expense of producing more noisy examples than L-BFGS. DeepFool [16] is an un-targeted adversarial attack method that produces adversarial examples containing less noise than L-BFGS, by approximating the decision boundaries of deep neural

networks with linear/affine classifiers, of the form $g(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$. The minimum perturbation $\boldsymbol{n}$ required to change the classifier label is estimated by the orthogonal projection of the sample $\boldsymbol{x}$ to the closest decision boundary, namely $\boldsymbol{n} = -\frac{g(\boldsymbol{x})}{\|\boldsymbol{w}\|^2}\boldsymbol{w}$. An iterative optimization algorithm estimates this perturbation, as follows:

$$\begin{aligned} \min_{\boldsymbol{n}}: \quad & \|\boldsymbol{n}\|_2^2 \\ \text{s.t.}: \quad & g(\tilde{\boldsymbol{x}}) - \nabla g(\tilde{\boldsymbol{x}})^T\boldsymbol{n} = 0, \end{aligned} \tag{3}$$

until the noise $\boldsymbol{n}$ is strong enough to change the classifier label. The above defined optimization problem can be extended to the multiclass case [16].

One of the most powerful targeted attacks up to date, that have been found to be effective against Defensive Distillation [8], as well as a number of other defenses [9] is the Carlini-Wagner (C & W) attack [7]. Its optimization problem involves minimizing a generalized distance function $D(\boldsymbol{x}, \tilde{\boldsymbol{x}})$, e.g., the L_0, L_2 and L_∞ norms, subject to a generalized classification function such that $C(\tilde{\boldsymbol{x}}) = t$, where $t \neq y \in \mathcal{Y}$. Moreover, it is assumed that $C(\tilde{\boldsymbol{x}}) = t$ if and only if an objective function $L_f(f(\tilde{\boldsymbol{x}}; \mathcal{W})) \leq 0$, where L_f may be a combination of different loss functions associated with f. This method is the generalization of the L-BFGS attack, having investigated different combinations of loss functions, suitable image data mappings for avoiding limitations of the box constraint $\boldsymbol{x}, \tilde{\boldsymbol{x}} \in [0, 1]$ and various gradient descend optimization algorithms.

Finally, we should also mention that other adversarial attack types have been proposed, such as the Jacobian-based Saliency Map Attack [17], or even attacks that modify only a single image pixel [18]. The reader is referred to the review papers [19, 20, 21] for more information.

2.2. Adversarial Defenses

Adversarial defenses are methodologies to protect against adversarial attacks. They focus on optimizing for achieving one of the two objectives:

- Adversarial attacks fail to deceive the defended model, i.e., $argmax(f(\tilde{\boldsymbol{x}}; \tilde{\mathcal{W}})) = y$, where $\tilde{\mathcal{W}}$ contains the weights of an appropriately modified deep neural network.
- The noise $\|\boldsymbol{n}\|_2$ energy required to be added to standard examples in order to deceive the classifier is increased beyond a level T, after a defense has been applied, i.e., $\|\boldsymbol{n}\|_2 > T$, so that $argmax(f(\boldsymbol{x} + \boldsymbol{n}; \tilde{\mathcal{W}})) \neq y$.

Some adversarial defense methods modify the neural network architecture by adding filtering/transformation layers before the neural network input layer [22, 23]. A recently proposed method [24] defines an adversarial prototype over an outer convex bound of a fixed size L_∞ ball, limited by the application of Rectified Linear Unit (ReLU) neural networks. Therefore, the minimum error threshold required to fool the classifier is related to the volume of the L_∞ ball. Perhaps the most straightforward approach to protect a neural network classifier against adversarial attacks without modifying the back-bone architecture, is to train it using adversarial examples. That is, adversarial examples are crafted from the training samples using e.g., one of the above mentioned adversarial attack methods, and thereby used to fine-tune the model weights with these examples and their original labels [16]. However, this approach imposes a significantly increased training complexity, mostly related to crafting different adversarial examples in each training epoch. An alternative approach is the so-called adversarial training [2]. That is, in addition to the standard classification objective functions, an additional objective function inspired by adversarial attack objectives is employed for training the classification model. To this end, Fast Gradient Sign objectives have been employed in the following manner [2]:

$$\tilde{L_f}(f(\boldsymbol{x}; \mathcal{W}), y) = \alpha L_f(f(\boldsymbol{x}; \mathcal{W}), y) + (1-\alpha) L_f(f(\tilde{\boldsymbol{x}}; \mathcal{W}), y), \tag{4}$$

where $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \epsilon \cdot \text{sign}(\nabla L_f(f(\boldsymbol{x}; \mathcal{W}), y))$ is an adversarial example derived by employing the Fast Gradient Sign attack, and $0 \leq \alpha \leq 1$ is parameter that controls the amount of regularization introduced to the model by this attack, by determining the contribution of FGS as a regularizer. A fixed value $\alpha = 0.5$ within the training process could be considered as a

valid starting point before tuning this parameter, as it has been shown to provide successful results [2]. This parameter could potentially be set to different values or even tuned in an adaptive fashion, considering that a value $\alpha = 1$ eliminates the regularization effect introduced by the second term, thus degenerating adversarial training to standard training. Adversarial training was proven to provide increased model resistance to the Fast Gradient Sign attack, in terms of the noise energy required to fool the model. As a side effect, the model generalizes better over unseen test examples. However, there is no guarantee that the final model is protected against all types of adversarial attacks. Other defense methods employ different adversarial training objectives that are more suitable for defending against other attacks [25, 26], or employ the standard adversarial training as a prepossesing step [9]. In fact, some of the most effective adversarial defenses up to date employ adversarial training using an ensemble of adversarial attacks methods, generated by diverse models [27].

An alternative approach that was initially found to dramatically decrease the success rate of adversarial attacks is to the so-called defensive distillation [8]. Distillation techniques [28] have been employed for knowledge transfer from a parent network to a distilled network having the exact same architecture. Depending on the value of so-called distillation temperature, the gradients of the distilled network vanish when calculated on the input samples, causing all gradient-based attacks to fail. However, it was shown that if an adversary is aware of the application of this defense, it is possible to estimate the distillation temperature using grid search [7]. This way, all defense operations can be reversed, and gradients can still be obtained for generating attacks [7]. In fact, it was later shown that a number of defenses, auto-encoder based, GAN-based, or even input manipulation-based defenses [29, 30, 31, 32, 33] ultimately rely on hiding/obfuscating the network gradients on the input samples, thus providing limited or no security gains against stronger attack methodologies [9, 34], or adversarial attacks exploiting the property of transferability.

3. Privacy protection against deep neural networks

Let $\mathcal{S} = \{\mathcal{X}, \mathcal{Y}\}$ be an image classification domain that a neural network classifier with trainable parameters $\mathcal{W}$ is very knowledgeable of, such that its decision function is able to map samples originating from this domain to their corresponding label space. That is, for any given sample $\boldsymbol{x} \in \mathcal{X}$, the neural network classifier is able to recover the label $y \in \mathcal{Y}$ by its decision function e.g., $argmax f(\boldsymbol{x}; \mathcal{W}) = y$. Adversarial attacks typically generate the perturbation as a mapping to a space of similar characteristics i.e., $\mathcal{X} \mapsto \tilde{\mathcal{X}}$ such that the ability of the classifier to map to the correct label is disabled $argmax f(\tilde{\boldsymbol{x}}; \mathcal{W}) \neq y$. It should be noted that this objective is typically achieved without taking into account any particular privacy protection constraints.

For instance, perhaps one of the most influential privacy protection concept is the K-Anonymity principles [35]. K-Anonymity is a generic privacy protection concept that suggests that the maximum possible probability of identifying an individual in a specific set must be lower than $1/K$ [35]. Adherence to this constraint can be achieved in many possible manners, depending on the data structure to be protected or the potential identifier. For example, in field structured data [36, 37, 38], the K-Anonymity concept is generalized as follows. It is assumed that an individual table record (e.g., record ID) may be identified by unique combinations of otherwise non distinguishing attribute tuples (e.g., sex, weight, age), the so-called Quasi-identifiers. Thus, dataset anonymization methods adhering to K-Anonymity principles, mask the appropriate attributes (e.g., table columns that act as Quasi-Identifiers) such that the remainder unique data combinations cannot be narrowed down to less than K combinations. That is, the maximum probability of identifying unique individuals cannot exceed $1/K$. In that field, the concept of K-Anonymity is achieved regardless of the discrimination abilities of the potential identifier, that can be either human or machine. K-Anonymity is related to data usability, since it essentially guides towards the relevant methods to solutions that achieve the K-Anonymity principles with the least possible data

manipulations.

In order to quantify privacy protection introduced by Adversarial Attacks according to the K-Anonymity principles, one might employ the class identification probabilities (e.g., classification rate) of a set of adversarial examples $\tilde{\mathcal{X}}$ against the classifier decision function $argmax f(\tilde{\boldsymbol{x}}; \mathcal{W}) = y$. However, this definition does not take into account the whole neural network output layer. More specifically, according to the perspectives of Label Ranking [39] or Multi-Label Classification [40], the output activation values of a deep neural network for sample $f(\boldsymbol{x}; \mathcal{W})$ encode an underlying strict ordered ranking $\succ_{\boldsymbol{x}} \subseteq \mathcal{Y} \times \mathcal{Y}$ over the finite label set $\mathcal{Y} = \{\ell_i, \ldots, \ell_C\}$, where $\ell_i \succ_{\boldsymbol{x}} \ell_j$ denotes that for a given data example $\boldsymbol{x}$, label ℓ_i is a more preferable output classification label than label ℓ_j. The ranking over $\mathcal{Y}$ is obtained by a unique permutation $\tau_{\boldsymbol{x}}(i) < \tau_{\boldsymbol{x}}(j)$ whenever $\ell_i \succ_{\boldsymbol{x}} \ell_j$, i.e., $\tau_{\boldsymbol{x}}(i)$ denotes the position of ℓ_i in the ranking. For simplicity reasons, we denote the label ranked at position i in the permutation with $r_{\boldsymbol{x}}(i)$, such that the output classification label of sample $\boldsymbol{x}$ by the deep neural network model is given by $argmax(f(\boldsymbol{x}; \mathcal{W})) = r_{\boldsymbol{x}}(1)$. We argue that the ranking obtained for any sample $\boldsymbol{x}$ encodes underlying data properties, that may be considered as Quasi-identifiers to the class of interest.

Therefore, in order to preserve anonymity (in the sense of hiding the true label) of every sample $\boldsymbol{x} \in \mathcal{X}$ against the neural network, inspired by the K-Anonymity principles, we argue that the appropriate mapping $\mathcal{X} \mapsto \tilde{\mathcal{X}}$ should achieve two conditions:

$$r_{\tilde{\boldsymbol{x}}}(1) \neq y, \quad \forall \tilde{\boldsymbol{x}} \in \tilde{\mathcal{X}}, \tag{5}$$

$$p(i) = \begin{cases} P\left(r_{\tilde{\boldsymbol{x}}}(i) = y\right) \leq 1/K & , \forall i \in \{1, \ldots, C\} \\ 0 & , \text{otherwise,} \end{cases} \tag{6}$$

where $p(\cdot)$ is the probability mass function of its argument, and K is a variable denoting the $K-$anonymity protection level, e.g., 5-Anonymity. Condition (5) is the Adversarial Attack objective, i.e., disabling correct classification. This constraint is commonly satisfied by almost every adversarial attack method. Condition (6) is the novel K-Anonymity-inspired objective, that achieves anonymity in adversarial examples along the whole network output. Without it, we argue that the network may still be used to classify adversarial examples by exploiting rule-based reasoning, e.g., by exploiting an adversarial example detector in the system. For instance, if an adversarial attack has been detected, the example may still be classified correctly using the same network, only using the output of e.g., the 2nd ranking position $r_{\tilde{\boldsymbol{x}}}(2)$ instead of the 1st $r_{\tilde{\boldsymbol{x}}}(1)$. Condition (6) guarantees that such simplified reasoning rules are impossible to be devised for the adversarial examples, exploiting the K-Anonymity concepts. In the next subsection, we describe a methodology that achieves both objectives (5) and (6) at the same time.

3.1. K-Anonymity-inspired Adversarial Attack

The proposed K-A^3 aims to generate the minimum required perturbations $\boldsymbol{n}_i$ to be added to samples originating from domain $\mathcal{S}$ in order to form a set of adversarial examples $\tilde{\mathcal{X}} = \{\tilde{\boldsymbol{x}}_i\}_{i=1}^N, \tilde{\boldsymbol{x}}_i = \boldsymbol{x}_i + \boldsymbol{n}_i$. To this end, along with optimizing for the adversarial attack objective (i.e., fooling the classifier decision function) (5), it also requires that the adversarial sample labels cannot be recovered by some specific sorted ranking position (6), inspired by the K-Anonymity principles. That is, the originating labels of the derived adversarial set $\tilde{\mathcal{X}}$ must recovered by at least K different positions in the sorted rankings, with probability no more than $p(i) \leq 1/K, i = 1, \ldots, K$ at each position.

Due to the increased optimization demands of the proposed K-A^3, it can be expected that increased perturbation will be generated to the crafted adversarial examples. To counteract this effect, we also introduce a similarity-based loss function $s(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ between the initial sample and the crafted adversarial example, guiding the optimization problem towards solutions that regulate the amount of noise generated by the adversarial attack, according to some objective metric. The CW-SSIM metric [15] is employed to this end. Thus, we introduce an additional constraint $d - s(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ to the proposed objective function to be minimized, where $d = max(s)$ (i.e., $d = 1$ for the CW-SSIM case).

Without violating the constraints (5) and (6), we demand that the actual labels of exactly K data

groups, each containing N/K samples of dataset $\tilde{\mathcal{X}}$, cannot be retrieved by in at least $k \in \mathcal{K} = \{2, \ldots, K+1\}$ sorted ranking positions, relevant to K. Assuming $K = 5$, then 5 data groups are formed, demanding that the actual labels of the first group cannot be retrieved by any of the ranking positions $r_{\boldsymbol{x}}(1), r_{\boldsymbol{x}}(2)$, while the labels of the second set are not retrieved in $r_{\boldsymbol{x}}(1), r_{\boldsymbol{x}}(2), r_{\boldsymbol{x}}(3)$ etc., while the labels in the $5-$th group are not retrieved in any position of $r_{\boldsymbol{x}}(i), \forall i \leq 6$. Maintaining the assumptions of white-box attacks i.e., access to a continuous loss function L_f associated with f, we propose the following optimization problem:

$$\min_{\boldsymbol{n}}: \quad \|\boldsymbol{n}\|_2 + (d - s(\boldsymbol{x}, \tilde{\boldsymbol{x}})) + \sum_{i=2}^{k} L_f(f(\tilde{\boldsymbol{x}}; \mathcal{W}), r_{(i)})), \tag{7}$$

until the output ranking obtained for $\tilde{\boldsymbol{x}}$ by the neural network architecture satisfies the constraint $r_{\tilde{x}}(i) \succ_{\tilde{\boldsymbol{x}}} y, \forall i \in \mathcal{K}$[1]. In fact, instead of using the ranked label positions, any k randomly selected target labels $\ell_i \neq y \in \mathcal{Y}$ could be employed in the proposed method, as well, without violating the constraint (6). However, it should be also be noted that the variable $k \in \mathcal{K}$ must be set to different value for every N/K sample groups, (e.g., $k = 2$ for group 1, $k = K+1$ for group K). For instance, if the variable k is set equal to some specific value of $\mathcal{K}$ for every of the N adversarial examples to be crafted (e.g., $k = 5$), then the constraint (6) will be violated, since $P(r_{\tilde{x}}(5) = y) > 1/K$.

The value of K should be carefully chosen by a potential user, depending on the K-Anonymization needs at hand. Although there is no optimal value setting, there are some trade-offs between different values of K that should taken into consideration. Larger values of K indicate the requirement of more confusing adversarial examples for the task classifier, hence stronger attacks. In that sense, K-A^3 attacks come at the expense of producing more noisy adversarial examples, and perhaps accompanied with increased difficulty in convergence, thus slower adversarial example production speed. Finally, as can be observed in (7), for a given $K = 1$ (i.e., $k = 2$ for all training data) and by omitting the visual similarity term, the proposed method degenerates to the standard L-BFGS method [1], with the only difference being that in L-BFGS, the target label t is selected by the adversary, instead of using the label retrieved in the 2nd sorted ranking position $r_{\tilde{x}}(2)$. This is also the case in two-class classification problems. Therefore, K-A^3 method can be viewed as a generalization of L-BFGS, that respects and supports contraints inspired by the K-Anonymity principles, for the multi-class classification case.

[1]We assume that the network classification function does not misclassify any of the initial samples $\boldsymbol{x}$ used to craft the adversarial examples, i.e., $r_{\boldsymbol{x}}(1) = y$ and $r_{\boldsymbol{x}}(i) \neq y, \forall\, i \in \mathcal{K}$ for every $\boldsymbol{x}, y \in \mathcal{S}$.

4. Multiple SVDD Defense

The proposed M-SVDD-D method assumes having an undefended pretrained multi-class deep neural network architecture with a parameter set $\mathcal{W}$, consisting of L layers, where its final layer $\boldsymbol{W}^L \in \mathbb{R}^C$ involves inference with a linear multiclass classifier layer, supporting C classes. Our defense strategy involves creating a modified architecture $\tilde{\mathcal{W}}$, by replacing this linear classifier layer with C non-linear one-class classifiers, based on the SVDD method [14]. Each one-class classifier acts as a validator for the inputs belonging to each of the C classes, while one additional class is added to the model, for classifying all unvalidated input (adversarial class). The proposed defense architecture is depicted in Figure 1.

The novel SVDD classifiers operate in a space of arbitrary dimensionality, which is approximated by a subspace defined by trainable random projections. After training each SVDD classifier, we develop a multi-class classifier exploiting the outputs of every SVDD model. Subsection 4.1 describes the proposed novel variant that exploits negative examples in the standard SVDD optimization process and it is solved in its primal form. Subsection 4.2 described the additional novel components and properties that comprise the proposed adversarial defense framework.

4.1. SVDD exploiting negative examples

The standard kernel SVDD method [14] aims at generating the minimum bounding hypersphere in

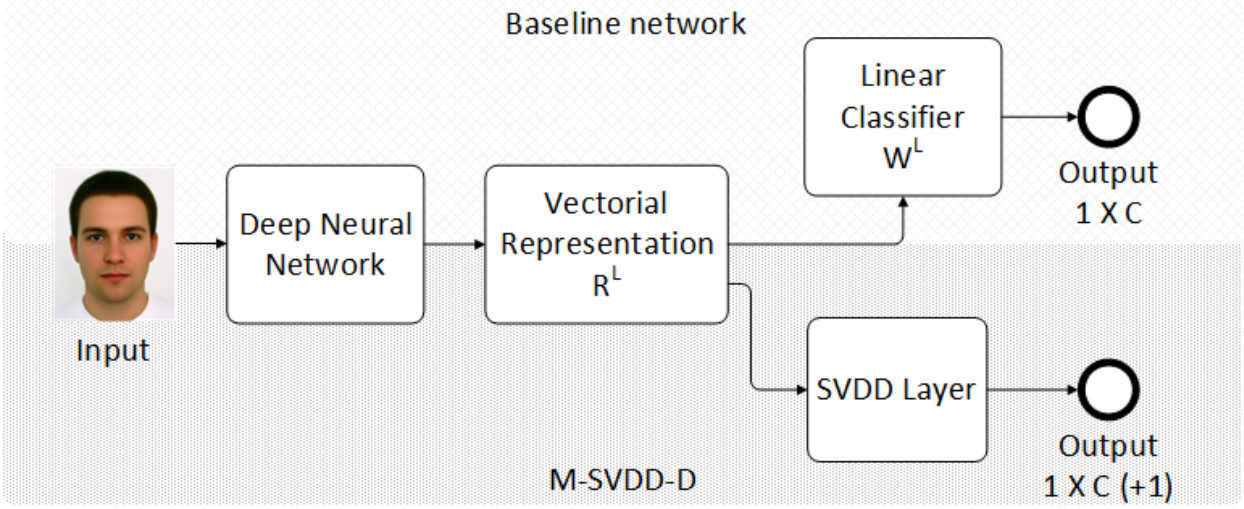


Figure 1: Conceptual diagram of the proposed M-SVDD-D. The baseline linear classifier layer is replaced by the proposed SVDD Layer. If the given input is not verified by any of the SVDD classifiers, it is classified to the adversarial class (C+1).

a space $\mathcal{H}$ of arbitrary dimensionality, having center $\mathbf{a} \in \mathcal{H}$ and radius R, which encloses the target class training vectors. This methodology has been extended to support training from negative [41] or unlabeled examples [42]. In this Section, we refer to the input data representations used to train the SVDD classifier with the notation $\boldsymbol{x} \in \mathbb{R}^l$, which indicate vectorial input data representations, for readability purposes.

Let $\boldsymbol{x}_p, p = 1, \ldots, N_p$ form a set of positive data (belonging to the target class), and $\boldsymbol{x}_n, n = 1, \ldots, N_n$ a set of negative data, respectively, employed to train the SVDD classifier. We assume that all training data vectors have been mapped to the space of arbitrary dimensionality by a function $\phi(\cdot) : R^l \mapsto \mathcal{H}$, using e.g., the RBF kernel function, introducing additional non-linearity, regardless of the employed network architecture. The primal SVDD optimization problem, exploiting negative examples in its optimization problem is defined as follows [14]:

$$\begin{aligned} \min_{R,\xi_p,\xi_n,\boldsymbol{a}}: \quad & R^2 + c_p \sum_{i=1}^{N_p} \xi_i + c_n \sum_{j=1}^{N_n} \xi_n \\ \text{s.t.:} \quad & \|\phi(\boldsymbol{x}_p) - \mathbf{a}\|^2 \leq R^2 + \xi_p, \\ & \|\phi(\boldsymbol{x}_n) - \mathbf{a}\|^2 > R^2 - \xi_n, \\ & \xi_p \geq 0, \xi_n \geq 0, \end{aligned} \tag{8}$$

where ξ_p, ξ_n are the slack variables and $c_p, c_n > 0$ are free parameters that allow some training error (i.e., soft margin formulation) for the positive and negative data respectively, in order to increase the generalization performance. After training, the decision value of the SVDD for a sample $\boldsymbol{x}$ can be obtained by:

$$g(\boldsymbol{x}) = R^2 - \|\phi(\boldsymbol{x}) - \mathbf{a}\|^2, \tag{9}$$

where $\boldsymbol{x}$ is classified to the positive class if $g(\boldsymbol{x}) \geq 0$ or considered as an outlier, otherwise.

Inspired by the standard SVDD method, we consider training an equivalent neural network with the same properties. To this end, employing traditional Quadratic Programming solvers is sub-optimal, especially when a large number of data is used for training

the network, since it would require massive amounts of memory (in order to store a kernel matrix of size $N \times N$, where N is equal to the number of data employed to train the network, while the network output is of significantly lower dimensionality $l \ll N$). We consider approximating the SVDD solution to this end. By exploiting the Representer Theorem [43], the hypersphere center $\mathbf{a} = \boldsymbol{\Phi}\boldsymbol{u}$ can be fully reconstructed by a matrix $\boldsymbol{\Phi} \in |\mathcal{F}| \times \mathbb{R}^N$ that contains the data representations in the arbitrary dimensional feature space, and an auxiliary vector $\boldsymbol{u}$ that contains its reconstruction weights. Since N is large, an approximation of the hypersphere center can be obtained by using a subset of the training data $\boldsymbol{o}_m \in \mathbb{R}^l$, $m = 1 \ldots, M$, $M \ll N$ [44], that are mapped in the feature space using the same function $\tilde{\boldsymbol{\Phi}} \in |\mathcal{F}| \times \mathbb{R}^M$, and the reconstruction vector $\tilde{\boldsymbol{u}} \in \mathbb{R}^M$.

Therefore, we devise a neural network architecture with trainable parameters $R, \boldsymbol{u}$ and $\boldsymbol{O}$, corresponding to a Radius R, a reconstruction vector $\boldsymbol{u}$ and a matrix $\boldsymbol{O}$, that contains trainable random projection vectors, initiated by forward-passing a subset of the training data from the neural architecture. Its decision function is defined as follows:

$$SVDD(\boldsymbol{x}) = \beta R^2 - k_{ii} + 2\tilde{\boldsymbol{k}}^T\boldsymbol{u} - \boldsymbol{u}^T\tilde{\boldsymbol{K}}\boldsymbol{u}, \quad (10)$$

where $k_{ii} = \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_i)$ is the output of the kernel function associated with $\mathcal{H}$ ($k_{ii} = 1$ in the RBF kernel case, for every training sample $\boldsymbol{x}$), $\tilde{\boldsymbol{k}} \in \mathbb{R}^{M\times 1}$ is a vector that contains data similarity between the training sample $\boldsymbol{x}_i$ and the random projections $\boldsymbol{O}$, and finally $\tilde{\boldsymbol{K}} \in \mathbb{R}^{M\times M}$ is the kernel matrix of the random projection vectors. Since the SVDD models will be used to defend against adversarial examples, an additional parameter $\beta > 0$ has been introduced to manually adjust the classifier precision ($\beta = 1$ is assumed during training process).

By incorporating the constraints of (8), we propose the following hinge loss functions to be associated with the $SVDD(\boldsymbol{x})$, that can be employed to approximate the solution of the SVDD optimization problem in its primal form:

$$L_p = max(0, k_{pp} - 2\tilde{\boldsymbol{k}}^T\boldsymbol{u} + \boldsymbol{u}^T\tilde{\boldsymbol{K}}\boldsymbol{u} - R^2), \quad (11)$$

$$L_n = max(0, R^2 - k_{nn} + 2\tilde{\boldsymbol{k}}^T\boldsymbol{u} - \boldsymbol{u}^T\tilde{\boldsymbol{K}}\boldsymbol{u}), \quad (12)$$

for the positive and the the negative data, respectively. The losses generated by the inference of positive and negative training samples to the SVDD model can thereby be used for back-propagating gradients to update the values of $R, \boldsymbol{u}$ and $\boldsymbol{O}$.

Experimentally, we have found that each SVDD should be trained by employing only positive data for a number of epochs until it converges, using only (11) as loss function, before both losses are implemented, since the proposed loss functions produce opposite gradient directions for the hyperphere radius R and the reconstruction vector $\boldsymbol{u}$, e.g., positive loss L_p promotes to increase the current value of the hypershere radius, where negative loss L_n decreases it. After converging to some values, then both loss functions can be implemented at the same time.

4.2. M-SVDD Defense

Let $\tilde{\mathcal{W}}$ denote the weights of a modified neural network architecture, where its final layer has been replaced by the proposed SVDD layer, formed by the trained SVDD classifiers for each class. This architecture is the core component of the proposed adversarial defense method. The outputs obtained by each SVDD classifier are top-bounded to $(-\infty, R^2]$, where a value of $SVDD(W(\boldsymbol{x})) = R^2$ means that the vectorial representation of the sample $\boldsymbol{x}$ obtained by forward passing $\mathcal{W}$, lies exactly at the learned hypersphere center. To create a multi-class classification model, the outputs of the SVDD layer are regularized by dividing with their respective radius values R_i^2 to $(-\infty, 1]$, corresponding to each class. Let $\boldsymbol{r} = [r_1, \ldots, r_C]^T$ be a vector that contains the regularized SVDD layer responses for a sample $\boldsymbol{x}$, $r_i = SVDD_i(W(\boldsymbol{x}))/R_i^2, i = 1, \ldots, C$, corresponding to each class. Then, sample $\boldsymbol{x}$ can be classified using a standard multi-class decision function $f(\tilde{\mathcal{W}}; \boldsymbol{x}) = argmax(\boldsymbol{r})$.

In order to improve inference performance, the modified architecture weights $\tilde{\mathcal{W}}$ are thereby fine-tuned for some epochs using the same training samples that have been employed for obtaining $\mathcal{W}$, by freezing the weights of the SVDD layer parameters. Since this model is multiclass, a suitable loss function (e.g., Cross-Entropy loss) associated with f is employed to this end. This training/fine-tuning proce-

dure introduces additional non-linearity to the architecture, imposed by the SVDD layer. Moreover, after convergence, we slightly modify the architecture so as to add a ReLU function $r = \max(0, SVDD(W(\boldsymbol{x}))$ right before the SVDD layer, and an additional clamping function $\min(v_p, r), v_p > 0$ right after it. This procedure obfuscates gradient generation[2] for any given input, since the decision values for a given sample $\boldsymbol{x}$ of the defense architecture is now quantized $r_i \in \{0, v_p\}, i = 1, \ldots, C$.

Finally, an additional input verification mechanism is included. That is, all possible inputs to the model are initially assumed to be adversarial examples unless verified by the respective SVDD classifier. One additional class $C + 1$ (adversarial class) is added to the proposed multi-class classification model, having a pre-specified output $A(\boldsymbol{x}) = v_a$ for every possible input, where $0 < v_a < v_p$. Parameter v_a essentially introduces the concept of *minimum acceptable activation value* for a given input sample. Samples failing to pass this threshold, are automatically classified to the adversarial class. In our experiments, we have set the minimum activation value $v_a = 0.001$ with successful results. This parameter may potentially be set to larger/smaller values, tuning the proposed M-SVDD Defense sensitivity to adversarial attacks.

The output classification label for sample $\boldsymbol{x}$ including the input verification mechanism is given by:

$$f(\tilde{\mathcal{W}}; \boldsymbol{x}) = argmax(\tilde{\boldsymbol{r}}), \tag{13}$$

where $\tilde{\boldsymbol{r}} = [r_1, \ldots, r_C, v_a]^T$ is the modified response vector, that supports C classes and one adversarial class. A sample $\boldsymbol{x}$ is classified to class c if the c-th SVDD classifier is its unique verifier, or considered as an adversarial example, otherwise. Here it should be noted, that instead of a-priory assuming that every given input imposes potential threats (i.e., belongs to the adversarial class), an adversarial example detector could be used instead [45, 46], in order to trigger this mechanism.

The final step of the defensive architecture training process includes optimizing the parameter β defined in equation (10) for each SVDD model, in order to trim the learned hypersphere volumes, inducing a more strict verification process, at the expense of reducing the architecture generalization performance for non-adversarial examples.

[2]This assumption holds only for the cases that the attacker has no access to the source code of the defense architecture, in order to remove these protective layers.

5. Experiments

In order to prove the concepts and evaluate the performance of the proposed methods, we have performed 2 sets of experiments, corresponding to the adversarial attack and adversarial defense scenarios. We have employed 4 publicly available image classification datasets, namely the MNIST (digit classification) [47], CIFAR-10 (object recognition) [48], Yale (face recognition) [49] and GTS (Traffic Sign Classification) [50] datasets, consisting of $70000, 60000$, 2452 and 39270 items, respectively. A total of 6 different architectures were trained until convergence in the employed datasets. Three architectures were employed in MNIST dataset, a) a four layer neural architecture consisting of fully connected layers and rectified linear units (MNIST-FC1), having an output dimensionality $l = 100$ and supporting 10 classes, b) a CNN architecture, namely the LeNet5 (MNIST-LeNet) [47] and finally, c) a recently proposed defensive architecture [24], that has been especially trained to increase the network robustness against L_2 adversarial perturbations (such as most of the employed adversarial attacks, including the proposed ones). In CIFAR-10 dataset, we have trained the MobileNetV2 architecture [51] (CIFAR-10-MobileNetV2). In Yale dataset, we fine-tuned a 9-Layer LightCNN architecture [52], that had been pre-trained using more than 1.5M facial images from CelebA dataset (Yale-LightCNN). Finally, for GTS dataset, we have fine-tuned an Imagenet pre-trained Resnet18 architecture [53], totaling 5 architecture-dataset combinations. All conducted experiments were implemented in PyTorch v0.4.

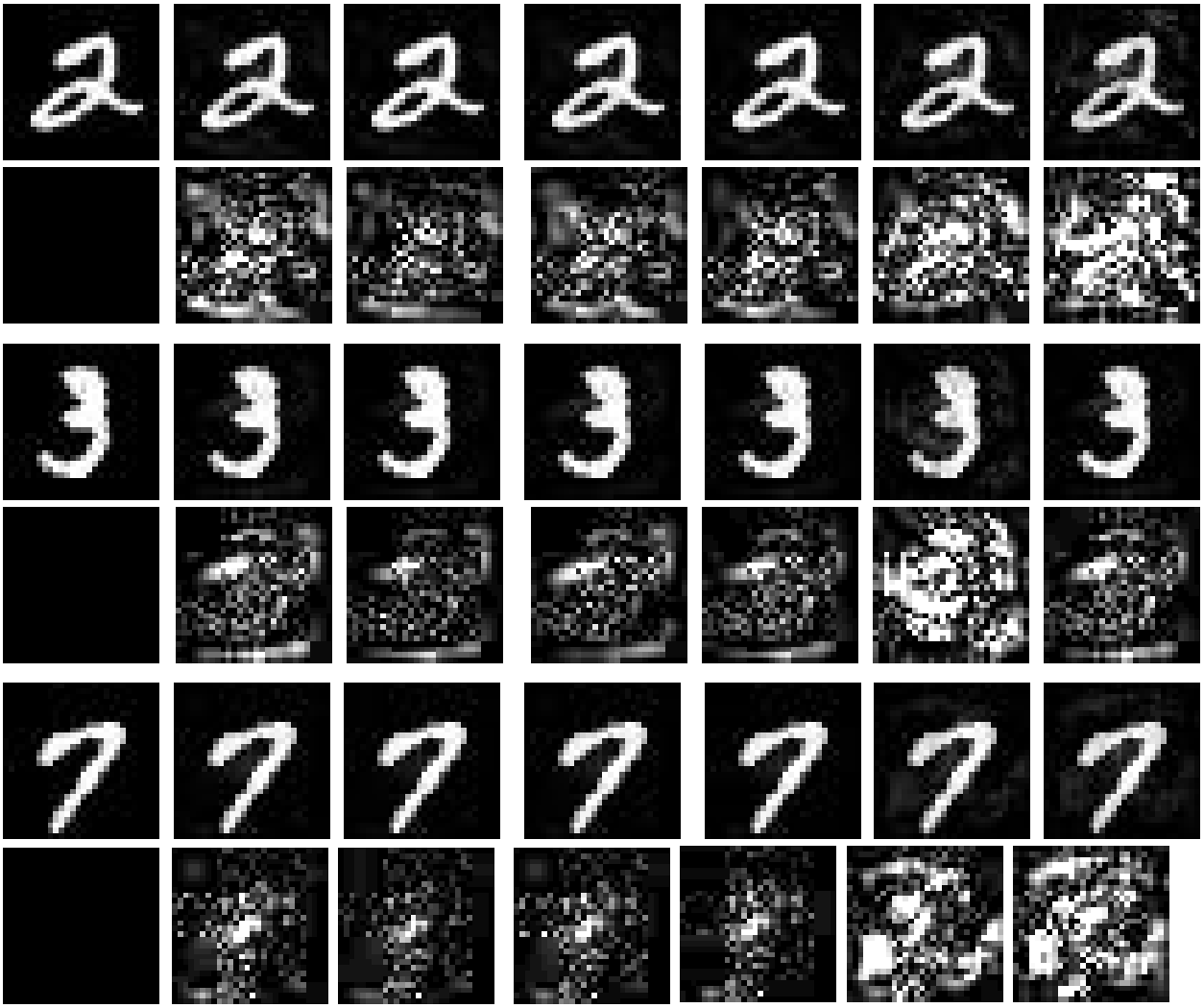
In our first set of experiments, we evaluate the performance of the proposed K$-A^3$ method. The proposed method was employed to attack each architecture for different values of $K = 1, 5, 9$. For comparison reasons, we have also employed the L-BFGS

[1], DeepFool [16] and the C & W [54] attack with L_2 distance. All methods were implemented using their default parameter settings. In the optimization process, we have slightly tuned the learning rate parameter of the optimizers (ADAM [55], SGD) in each experiment, while keeping their settings equal for all competing methods. The same target class labels were assigned to L-BFGS and C & W attacks. We have employed these methods to generate adversarial datasets $\tilde{\mathcal{X}}$ by modifying the training samples of each dataset. Adversarial examples generated by the competing methods in (MNIST-FC1) and Yale-LightCNN, as well as indicative images containing the perturbation generated by each method, are shown in Figures 2 and 3. As can be observed, the adversarial attacks of almost all methods are hardly perceptible to the human eye, producing useful images that can not be classified correctly by the respective classifier.

The datasets obtained by each method were evaluated in terms of satisfying "the ranking anonymization properties", as have been defined in equation (6) of this paper. That is, we have tried to retrieve the original dataset labels using the architecture $\mathcal{W}$ to obtain ranked label outputs for each adversarial sample. We have determined the probability mass functions $p(i)$ for obtaining the ground truth label at the $i-$th ranking position, plotted in Figure 4. As can be observed, the datasets obtained by employing L-BFGS, DeepFool, C & W, and the variant 1-A^3 of the proposed method do not satisfy the constraint (6), since $P(r_{\tilde{\boldsymbol{x}}}(2) = y) > 1/K$ for every $K > 1$. On the other hand, the probabilities of the adversarial examples crafted by the proposed 5-A^3 and 9-A^3 methods, satisfy $P(r_{\tilde{\boldsymbol{x}}}(j) = y) \leq 1/5$ for $i = 2, \ldots, 6$ and $P(r_{\tilde{\boldsymbol{x}}}(j) = y) \leq 1/9$ for $j = 2, \ldots, 10$ respectively in almost every case, or lie really close.

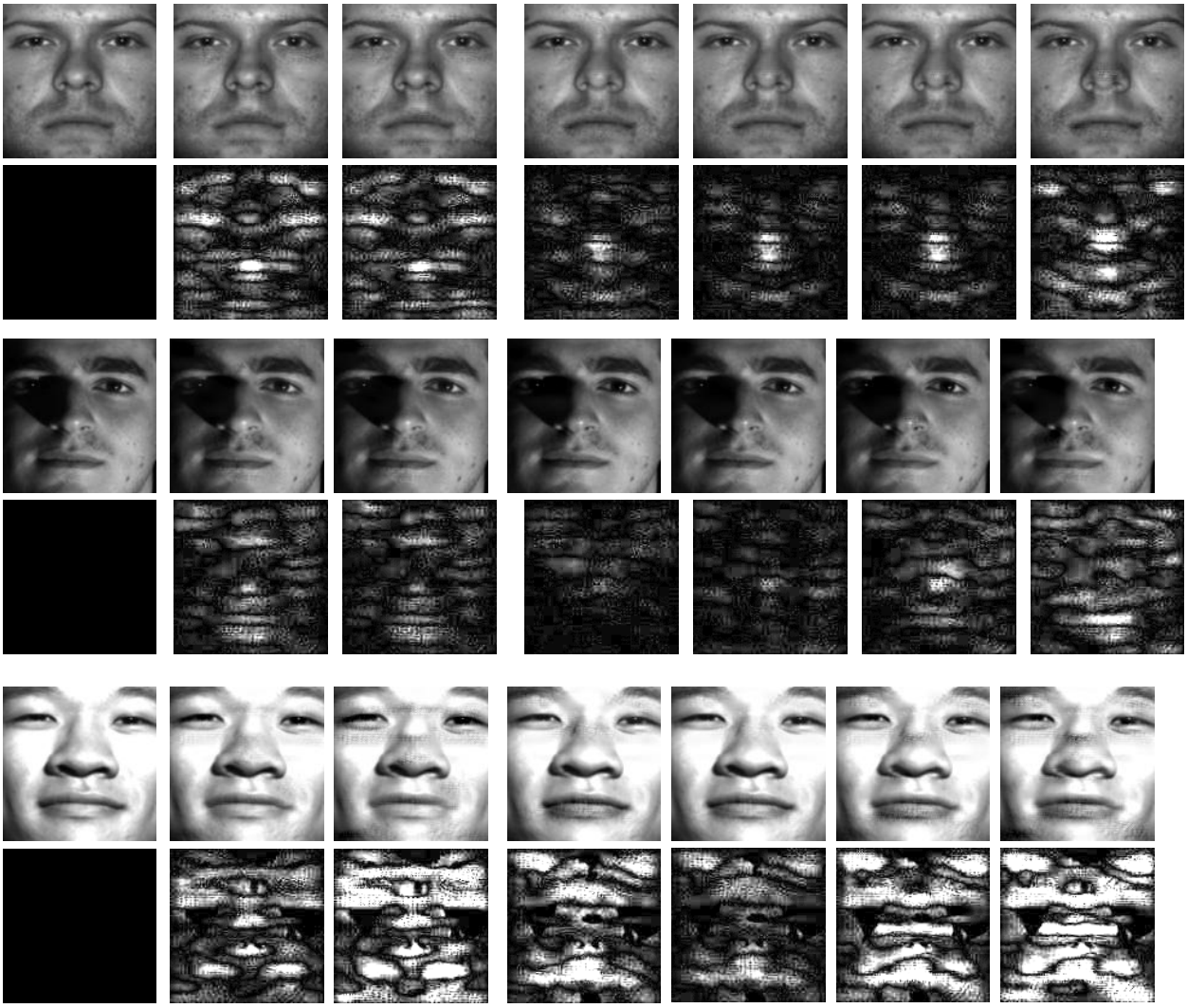
In addition, all adversarial attack methods were evaluated in terms of the introduced perturbation. As evaluation metrics, we computed the average Mean Squared Error (MSE)= $\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_2^2$ and average Structural Similarity [15] (SSIM) for the adversarial datasets crafted by each method. Reported SSIM values were scaled from $[0, 1]$ to $[0, 100]$, for readability purposes. In CIFAR-10-MobileNet and GTS-Resnet, the Multi-Scale Structural Similarity (MS-SSIM) index is reported instead of SSIM, which is more suitable for RGB images. Higher MSE values denote that the adversarial examples contain increased perturbation, and high SSIM values denote that the adversarial example $\tilde{\boldsymbol{x}}$ appears visually similar with the training example $\boldsymbol{x}$. Results of the evaluation are drawn on the left side of Tables 2, 3 and 4, under the "Undefended" category. The proposed $1 - A^3$ generated the least amount of perturbation and generated the most visually similar examples, in the light of the selected evaluation metrics, especially when compared to the standard L-BFGS attack, attributed to employing the SSIM loss in its optimization process. The introduced perturbation of the proposed $5 - A^3$ and $9 - A^3$ was increased when compared to $1 - A^3$. This was expected due to the demand of adherence to the K-Anonymity-inspired constraint. However, it should be noted that the visual similarity of the adversarial examples generated by the proposed $5 - A^3$ and $9 - A^3$ with the original images, have been found to be very close, or even increased in some cases, when compared to the similarity of adversarial examples generated by L-BFGS with the original images. This effect should also be attributed to the exploitation of the SSIM loss function.

In our second set of experiments, we evaluated the proposed M-SVDD-D method. We have trained one SVDD-based classifier for each class by exploiting the pretrained architectures and the training data samples, for $20 - 50$ epochs. The initial weights of the architectures remained fixed during the SVDD classifier training procedure. The number of trainable random projections M was set to 512 in MNIST-FC1, 256 in MNIST-LeNet and MNIST-PR, and 128 in CIFAR10-MobileNetV2, Yale-LightCNN and GTS-Resnet, subject to computational and memory constraints, imposed by the size of the employed network architectures, as well as the input image size. Then, we have created modified architectures $\mathcal{W}$ by replacing the linear classifier layer, with the trained SVDD classifiers (SVDD layer). We have fine-tuned the network architectures with the SVDD layer for an additional 15 epochs, by fixing the SVDD classifier parameters, this time. Here, it should be noted that the initial linear classifier layer may be re-attached to the network architecture, without affecting its generalization performance, or even increasing it in some cases,



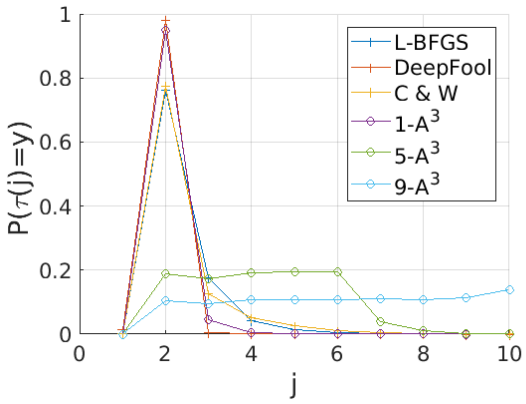
(a) $X - X$ (b) FGS (c) C&W (d) DFool (e) 1-A^3 (f) 5-A^3 (g) 9-A^3

Figure 2: Adversarial examples on MNIST Dataset. On the first row, top left (a) depicts the original image $\boldsymbol{x}$, (b)-(g) are the corresponding adversarial examples $\tilde{x}$ derived by L-BFGS (b), C&W (c), DeepFool (d) and the proposed methods (e)-(g). The second row depict the perturbation generated by each method, magnified 10 times for visibility purposes, i.e.,(a) $\|\boldsymbol{x} - \boldsymbol{x}\|$ and (b)-(g) $\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|$. The pattern is repeated for different samples in rows 3-4 and 5-6, respectively.

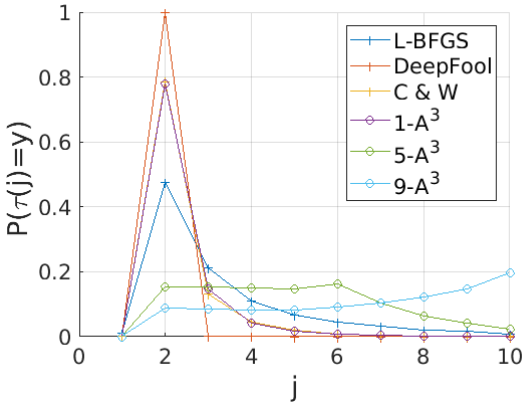


(a) X (b) FGS (c) C&W (d) DFool (e) 1-A^3 (f) 5-A^3 (g) 9-A^3

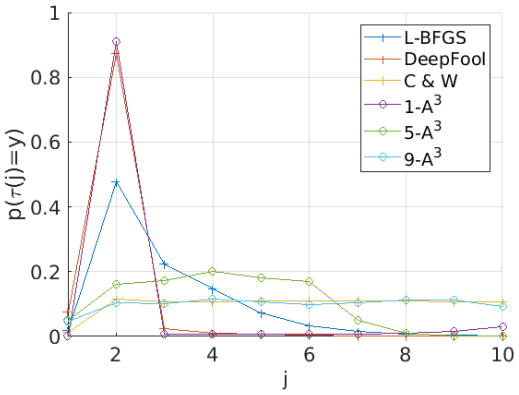
Figure 3: Adversarial examples on Yale Dataset. Top left (a) depicts the original image, (b)-(g) are the adversarial examples corresponding to each method. The second row depict the perturbation generated by each method (magnified by a scale of 10).



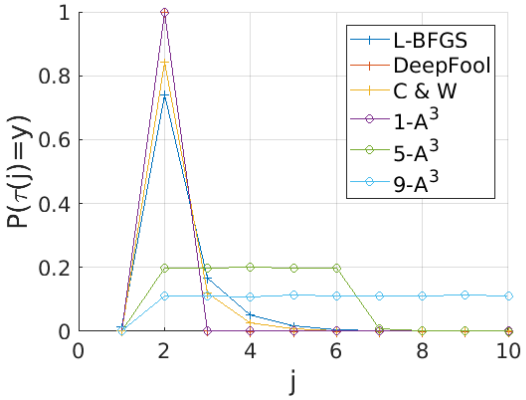
(a) MNIST-FC1



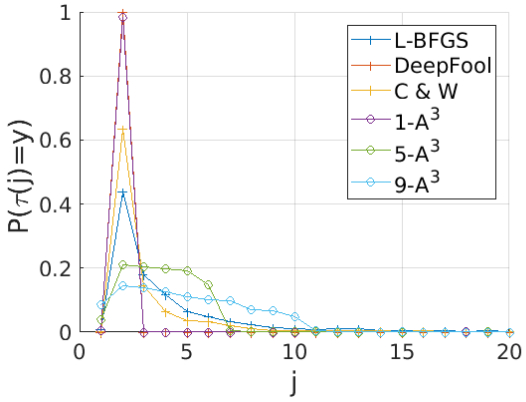
(b) MNIST-LeNet



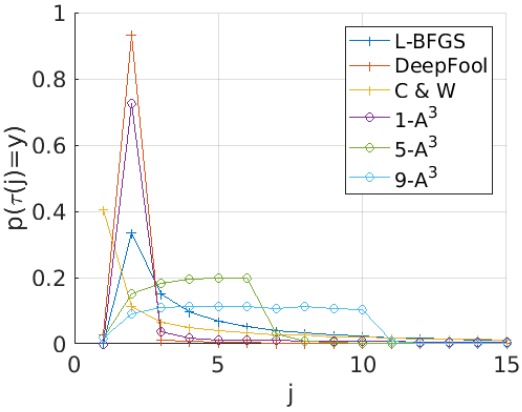
(c) MNIST-PR



(d) CIFAR-MobileNetV2



(e) Yale-LightCNN



(f) GTS-ResNet

Figure 4: Probability mass functions of recovering the original labels y in the j-th sorted ranking position $P(r_{\tilde{\mathbf{x}}}(j) = y)$ obtained by exploiting the undefended architecture $\mathcal{W}$, for each adversarial dataset $\tilde{\mathcal{X}}$ generated by L-BFGS, DeepFool, C & W, and the proposed methods, corresponding to each image classification dataset. As can be seen, L-BFGS, DeepFool, C & W, and the variant 1-A^3 of the proposed method do not satisfy the K-Anonymity-inspired objective (6), since in most cases, the original label y can be recovered by retrieving the label ranked 2nd.

as shown in Table 1. For instance, in MNIST-PR, since the model was already optimized for defense purposes, the application of the proposed method increased its generalization performance significantly.

Table 1 reports the test accuracy of the employed architectures, using the following settings. $\mathcal{W}$-LC refers to the default architecture before the application of SVDD, $\tilde{\mathcal{W}}$-LC refers to the modified architecture, having re-attached the Linear Classifier Layer, $\tilde{\mathcal{W}}$-SVDD($\beta = O_c$) refers to the defense architecture where all parameters β are have been optimized for classification purposes, and finally $\tilde{\mathcal{W}}$-SVDD($\beta = O_d$) refers to the architecture where the parameters β have been optimized for defense purposes. As can be observed, the modified architecture $\tilde{\mathcal{W}}$-LC does not have reduced classification accuracy, when compared to $\mathcal{W}$-LC. This could potentially mean that the use of SVDD classifiers may act as regularizer to the network parameters, thus providing increased neural network generalization abilities, but this statement has to be confirmed in a different evaluation scenario, which is outside of the scope of this work. In almost every case, the performance of the M-SVDD network, especially for the cases where parameters β have been optimized for classification purposes (i.e., $\beta = O_c$), closely matches the performance of the standard architecture.

We have employed the adversarial attack methods to attack the M-SVDD-D defense architectures, by exploiting the transferability property of adversarial attacks, using $\tilde{\mathcal{W}}$-LC as the intermediate architecture (white-box attack). We assumed that a potential attacker has access to the original architecture and the modified weights by the proposed defense architecture with the linear classifier ($\tilde{\mathcal{W}}$-LC), while has no access to the source code of the SVDD models and the corresponding parameters. This scenario assumes that the proposed M-SVDD-D defense is employed as an additional input verification mechanism. Results are reported on the right side of Tables 2, 3 and 4, under the "defended" column. As can be observed, examples produced by $\tilde{\mathcal{W}}$-LC contain more perturbation for almost all the employed adversarial attack methods, when compared to the perturbations generated by $\mathcal{W}$-LC as depicted on the left columns, in terms of MSE and SSIM, in almost every case. The only case where $\tilde{\mathcal{W}}$-LC seemed to decrease the introduced perturbation is when the architecture was already optimized for defense purposes (i.e., MNIST-PR).

Table 1: Classification accuracy of the employed architectures

Architecture	MNIST-FC1	MNIST-LeNet	MNIST-PR	CIFAR-10-MobileNetV2	Yale-LightCNN	GTS-Resnet
$\mathcal{W}$-LC	**98.64**	**99.12**	87.29	87.28	97.25	93.10
$\tilde{\mathcal{W}}$-LC	98.32	**99.12**	94.48	**88.14**	**98.16**	**94.74**
$\tilde{\mathcal{W}}$-SVDD($\beta = O_c$)	98.14	96.62	**97.61**	87.47	96.33	82.19
$\tilde{\mathcal{W}}$-SVDD($\beta = O_d$)	95.19	95.00	93.47	86.36	93.68	77.25

Finally, we report the adversarial attack success rates (ASR%). Adversarial attacks are considered successful if $f(\tilde{\mathcal{W}}; \boldsymbol{x}) \neq y$, failed (F) if $f(\tilde{\mathcal{W}}; \boldsymbol{x}) = y$ and prevented/detected (D) if $f(\tilde{\mathcal{W}}; \boldsymbol{x}) = v_a$, where v_a denotes the label of the adversarial class. As can be observed, using a value of $\beta = 0.85$ leads to preventing over 95% of attacks in MNIST-FC1, while only sacrificing 3% of test accuracy. On the other hand, just the application of the M-SVDD-D leads to failure 48% of 1-A^3 attacks in Yale-LightCNN experiments. In almost every case, ASR rates drop significantly from the 100% on undefended architectures, when the M-SVDD-D is implemented. Even for the case where the architecture was already optimized for defense purposes (MNIST-PR), the proposed defense decreased the ASR% rates dramatically.

Therefore, M-SVDD-D is a promising adversarial defense mechanism. However, it should be noted that if an adversary is provided access to the SVDD layer source code, then the defense could be potentially reverse engineered. In such case, the success rates of his adversarial attacks against the defended network architecture will be similar to the undefended architecture, close to 100%, given appropriate parameter settings. However, it should also be expected that the generated adversarial examples will still contain similar perturbation with the one reported for the defended architecture in Tables 2, 3 and 4.

6. Conclusion

In this paper, an adversarial attack method was developed, that supports and respects the proposed K-Anonymity-inspired requirements. In addition, the defense mechanism proposed in this paper can be used to: a) increase adversarial attack failure rates, b) increase the noise energy required to be added to

Table 2: Experimental Results in MNIST dataset.

Experiment	**MNIST-FC1**						
	Undefended			Defended			
Attack Method/Metric	SSIM	MSE$\times 10^4$	ASR%	SSIM	MSE$\times 10^3$	ASR%, $\beta = 1$	ASR%, $\beta = 0.85$
L-BFGS	60.30	26.25	99.75	30.67	36.39	83.04 (F=06.21, D=10.74)	26.43 (F=00.02,D=73.54)
DeepFool	70.74	12.96	98.70	57.30	4.73	76.10 (F=22.57, D=01.32)	04.42 (F=00.00, D=95.56)
C & W	59.40	27.27	99.99	58.85	3.81	62.61 (F=21.96, D=15.42)	00.21 (F=02.89, D=96.89)
1-A^3	76.76	13.70	100	29.38	274.04	17.94 (F=05.31 D=76.74)	04.19 (F=0.00, D=95.80)
5-A^3	63.92	34.10	100	59.90	23.48	75.07 (F=12.34 D=12.58)	02.66 (F=0.00, D=97.31)
9-A^3	58.85	53.51	100	43.74	47.93	84.80 (F=12.68 D=02.50)	01.46 (F=0.00, D=98.53)
Experiment	**MNIST-LeNet**						
	Undefended			Defended			
Attack Method/Metric	SSIM	MSE$\times 10^3$	ASR%	SSIM	MSE$\times 10^3$	ASR%, $\beta = 1$	ASR%, $\beta = 0.97$
L-BFGS	73.75	2.42	99.98	38.30	10.56	72.59 (F=9.42, D=17.98)	54.09 (F=05.82, D=40.08)
DeepFool	80.35	1.57	99.92	78.36	1.71	33.93 (F=55.21, D=10.84)	19.91 (F=51.77, D=28.30)
C & W	80.12	1.45	99.93	79.91	1.34	35.70 (F=46.94, D=17.35)	20.24 (F=41.41, D=38.34)
1-A^3	84.64	1.87	100	45.35	8.15	78.17 (F=7.00, D=14.82)	61.69 (F=04.96, D=33.33)
5-A^3	72.72	5.37	100	50.86	13.56	63.67 (F=7.02, D=29.30)	39.29 (F=03.80, D=56.90)
9-A^3	61.69	15.15	100	44.77	23.25	61.37 (F=5.84, D=32.77)	32.27 (F=02.85, D=64.87)
Experiment	**MNIST-PR**						
	Undefended			Defended			
Attack Method/Metric	SSIM	MSE$\times 10^2$	ASR%	SSIM	MSE$\times 10^2$	ASR%, $\beta = 1.10$	ASR%, $\beta = 1$
L-BFGS	18.73	7.63	98.28	46.97	0.61	29.12 (F=06.60, D=64.25)	10.74 (F=01.62, D=87.63)
DeepFool	82.25	0.56	92.46	73.63	0.12	03.59 (F=63.06, D=33.33)	00.32 (F=28.61, D=71.07)
C & W	37.46	10.54	98.61	39.92	11.21	00.00 (F=00.00, D=100.0)	00.00 (F=00.00, D=100.0)
1-A^3	63.12	1.19	99.81	48.78	0.68	07.35 (F=04.55, D= 88.09)	01.75 (F=14.26, D=83.30)
5-A^3	41.88	4.10	95.13	45.68	0.77	14.80 (F=07.23, D=77.96)	03.91 (F=02.63, D=93.44)
9-A^3	38.08	4.75	95.20	48.78	0.27	13.74 (F=25.03, D=61.22)	02.43 (F=01.51, D=95.69)

Higher SSIM values indicate increased visual similarity between the crafted adversarial set and the initial set.
Higher MSE values are associated with increased perturbation.
"D" indicates Detected attacks and "F" Failed attacks.

Table 3: Experimental Results in CIFAR10-MobileNetV2 and Yale-LightCNN

Experiment	**CIFAR10-MobileNetV2**						
	Undefended			Defended			
Attack Method/Metric	MS-SSIM	MSE$\times 10^5$	ASR%	MS-SSIM	MSE$\times 10^4$	ASR%, $\beta = 1$	ASR%, $\beta = 0.95$
L-BFGS	99.84	5.35	98.53	99.33	41.05	94.26 (F=03.41, D=02.32)	71.71 (F=00.28, D=26.99)
DeepFool	99.99	2.04	99.74	99.93	3.03	70.69 (F=28.93, D=00.37)	59.07 (F=05.98, D=34.93)
C & W	99.99	3.65	99.96	99.84	0.45	62.69 (F=27.65, D=09.38)	24.34 (F=03.18, D=72.47)
1-A^3	99.99	1.65	99.95	99.63	23.93	92.99 (F=01.40, D=05.60)	48.22 (F=00.09, D=51.67)
5-A^3	99.99	4.96	99.77	99.36	81.26	78.84 (F=00.60, D=20.54)	19.73 (F=00.03, D=80.22)
9-A^3	99.98	8.08	99.85	99.02	45.12	56.34 (F=00.32, D=43.33)	11.04 (F=00.03, D=88.91)
Experiment	**Yale-LightCNN**						
	Undefended			Defended			
Attack Method/Metric	SSIM	MSE$\times 10^4$	ASR%	SSIM	MSE$\times 10^4$	ASR%, $\beta = 1.04$	ASR% , $\beta = 1$
L-BFGS	93.89	5.97	99.23	93.30	6.26	43.44 (F =19.36, D=37.19)	10.96 (F=08.81, D=80.22)
DeepFool	97.87	1.71	100	97.77	1.94	41.49 (F=47.43, D=11.06)	17.36 (F=37.85, D=44.77)
C & W	94.21	5.16	99.94	92.82	5.59	29.91 (F=24.64, D=45.44)	06.91 (F=12.70, D=80.37)
1-A^3	98.05	1.59	99.43	98.06	1.63	41.59 (F=48.82, D=09.57)	18.08 (F=38.11, D=43.80)
5-A^3	95.17	7.52	96.26	94.55	7.74	42.82 (F=18.80, D=38.37)	11.52(F= 12.09, D=76.38)
9-A^3	93.17	4.36	91.34	92.52	5.98	29.50 (F=15.47, D=55.02)	06.40(F=08.65, D=84.93)

Higher SSIM values indicate increased visual similarity between the crafted adversarial set and the initial set.
Higher MSE values are associated with increased perturbation.
"D" indicates Detected attacks and "F" Failed attacks.

Table 4: Experimental Results in GTS-Resnet

Experiment	**GTS-Resnet**						
	Undefended			Defended			
Attack Method/Metric	MS-SSIM	MSE$\times 10^5$	ASR%	MS-SSIM	MSE$\times 10^5$	ASR%, $\beta = 1.01$	ASR%, $\beta = 1.00$
L-BFGS	99.59	4.90	99.72	98.82	6.75	79.48 (F=05.85, D=14.67)	49.98 (F=04.60, D=45.42)
DeepFool	99.73	0.20	98.97	98.96	0.45	61.39 (F=31.30, D=07.30)	34.70 (F=38.35, D=26.95)
C & W	98.64	8.74	59.63	96.24	211.37	52.88 (F=41.75, D=05.37)	16.20 (F=56.07, D=27.73)
1-A^3	99.91	0.79	99.72	99.16	3.81	81.53 (F=11.78, D=06.69)	47.92 (F=12.12, D=39.96)
5-A^3	99.65	3.87	98.13	99.14	4.34	84.09 (F=04.81, D=11.10)	54.05 (F=06.98, D=38.97)
9-A^3	99.48	6.04	98.31	98.88	6.79	82.87 (F=03.51, D=13.62)	50.83 (F=05.81, D=43.36)

Higher SSIM values indicate increased visual similarity between the crafted adversarial set and the initial set.
Higher MSE values are associated with increased perturbation.
"D" indicates Detected attacks and "F" Failed attacks.

standard examples in order to be deceived, c) prevent adversarial attacks. Moreover, it was shown that the optimization problem of the SVDD classifier can be effectively solved in its primal form, using both positive and negative examples. The solution in the arbitrary dimensional space is approximated by a Neural Network architecture and randomly projected vectors.

Future research may focus towards the implementation of K-A^3 objectives across other modalities (e.g., sound), or in different adversarial attack methodologies, for privacy protection. Moreover, findings from applying proposed SVDD layer for defending against adversarial attacks, suggest that this methodology could be used in a more generic Neural Network training process. For example, slight modifications of the methodology could inspire the development of novel loss functions based on SVDD or other one-class classifiers in multi-class and one-class classification tasks, as well as in retrieval tasks.

Acknowledgment

This work has received funding from the European Union's European Union Horizon 2020 research and innovation programme under grant agreement no 731667 (MULTIDRONE). This publication reflects only the authors' views. The European Commission is not responsible for any use that may be made of the information it contains.

References

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

[2] I. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in: International Conference on Learning Representations, 2015.
URL `http://arxiv.org/abs/1412.6572`

[3] A. Nguyen, J. Yosinski, J. Clune, Deep neural networks are easily fooled: High confidence predictions for unrecognizable images, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 427–436.

[4] N. Papernot, P. McDaniel, I. Goodfellow, Transferability in machine learning: from phenomena to black-box attacks using adversarial samples, arXiv preprint arXiv:1605.07277.

[5] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, P. Frossard, Universal adversarial perturbations, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2017, pp. 86–94.

[6] W. Zhou, X. Hou, Y. Chen, M. Tang, X. Huang, X. Gan, Y. Yang, Transferable adversarial perturbations, in: The European Conference on Computer Vision (ECCV), 2018.

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

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

[9] A. Athalye, N. Carlini, D. Wagner, Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples, in: J. Dy, A. Krause (Eds.), Proceedings of the 35th International Conference on Machine Learning, Vol. 80 of Proceedings of Machine Learning Research, PMLR, Stockholmsmässan, Stockholm Sweden, 2018, pp. 274–283.
URL http://proceedings.mlr.press/v80/athalye18a.html

[10] B. Batrinca, P. C. Treleaven, Social media analytics: a survey of techniques, tools and platforms, AI & SOCIETY 30 (1) (2015) 89–116. doi:10.1007/s00146-014-0549-4.
URL https://doi.org/10.1007/s00146-014-0549-4

[11] Y. Liu, W. Zhang, N. Yu, Protecting privacy in shared photos via adversarial examples based stealth, Security and Communication Networks 2017.

[12] S. J. Oh, M. Fritz, B. Schiele, Adversarial image perturbation for privacy protection–a game theory perspective, in: International Conference on Computer Vision (ICCV), 2017.

[13] I. Sikiric, T. Hrkac, K. Zoran Kalafatic, et al., I know that person: Generative full body and face de-identification of people in images, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2017, pp. 15–24.

[14] D. M. Tax, R. P. Duin, Support Vector Data Description, Machine learning 54 (1) (2004) 45–66.

[15] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE transactions on image processing 13 (4) (2004) 600–612.

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

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

[18] J. Su, D. V. Vargas, S. Kouichi, One pixel attack for fooling deep neural networks, arXiv preprint arXiv:1710.08864.

[19] X. Yuan, P. He, Q. Zhu, R. R. Bhat, X. Li, Adversarial examples: Attacks and defenses for deep learning, arXiv preprint arXiv:1712.07107.

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

[21] N. Akhtar, A. Mian, Threat of adversarial attacks on deep learning in computer vision: A survey, IEEE Access 6 (2018) 14410–14430. doi:10.1109/ACCESS.2018.2807385.

[22] U. Shaham, J. Garritano, Y. Yamada, E. Weinberger, A. Cloninger, X. Cheng, K. Stanton, Y. Kluger, Defending against adversarial images using basis functions transformations, arXiv preprint arXiv:1803.10840.

[23] N. Das, M. Shanbhogue, S.-T. Chen, F. Hohman, L. Chen, M. E. Kounavis, D. H. Chau, Keeping the bad guys out: Protecting and vaccinating deep learning with jpeg compression, arXiv preprint arXiv:1705.02900.

[24] E. Wong, F. Schmidt, J. H. Metzen, J. Z. Kolter, Scaling provable adversarial defenses, in: Advances in Neural Information Processing Systems, 2018, pp. 8400–8409.

[25] A. Kurakin, I. J. Goodfellow, S. Bengio, Adversarial machine learning at scale, in: International Conference on Learning Representations (ICLR), 2017.

[26] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, in: International Conference on Learning Representations (ICLR), 2018.
URL https://openreview.net/forum?id=rJzIBfZAb

[27] F. Tramer, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, P. McDaniel, Ensemble adversarial training: Attacks and defenses, in: International Conference on Learning Representations, 2018.
URL https://openreview.net/forum?id=rkZvSe-RZ

[28] G. E. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, Deep Learning and Representation Learning Workshop, Conference on Neural Information Processing Systems (NIPS) abs/1503.02531.

[29] P. Samangouei, M. Kabkab, R. Chellappa, Defense-GAN: Protecting classifiers against adversarial attacks using generative models, in: International Conference on Learning Representations, 2018.
URL https://openreview.net/forum?id=BkJ3ibb0-

[30] G. S. Dhillon, K. Azizzadenesheli, J. D. Bernstein, J. Kossaifi, A. Khanna, Z. C. Lipton, A. Anandkumar, Stochastic activation pruning for robust adversarial defense, in: International Conference on Learning Representations, 2018.
URL https://openreview.net/forum?id=H1uR4GZRZ

[31] C. Guo, M. Rana, M. Cisse, L. van der Maaten, Countering adversarial images using input transformations, in: International Conference on Learning Representations, 2018.
URL https://openreview.net/forum?id=SyJ7ClWCb

[32] A. Prakash, N. Moran, S. Garber, A. DiLillo, J. Storer, Deflecting adversarial attacks with pixel deflection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 8571–8580.

[33] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, J. Zhu, Defense against adversarial attacks using high-level representation guided denoiser, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 1778–1787.

[34] A. Athalye, N. Carlini, On the robustness of the cvpr 2018 white-box adversarial example defenses, arXiv preprint arXiv:1804.03286.

[35] L. Sweeney, k-anonymity: A model for protecting privacy, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10 (05) (2002) 557–570.

[36] A. Campan, T. M. Truta, Data and structural k-anonymity in social networks, in: F. Bonchi, E. Ferrari, W. Jiang, B. Malin (Eds.), Privacy, Security, and Trust in KDD, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 33–54.

[37] J. Domingo-Ferrer, D. Sanchez, J. Soria-Comas, Database Anonymization: Privacy Models, Data Utility, and Microaggregation-based Intermodel Connections, Morgan and Claypool, 2016.
URL https://ieeexplore.ieee.org/document/7385400

[38] J. Soria-Comas, J. Domingo-Ferrer, Big data privacy: Challenges to privacy principles and models, Data Science and Engineering 1 (1) (2016) 21–28. doi:10.1007/s41019-015-0001-x.
URL https://doi.org/10.1007/s41019-015-0001-x

[39] J. Fürnkranz, E. Hüllermeier, E. Loza Mencía, K. Brinker, Multilabel classification via calibrated label ranking, Machine Learning 73 (2) (2008) 133–153. doi:10.1007/s10994-008-5064-8.
URL https://doi.org/10.1007/s10994-008-5064-8

[40] J. Zhu, S. Liao, Z. Lei, S. Z. Li, Multi-label convolutional neural network based pedestrian

attribute classification, Image and Vision Computing 58 (2017) 224–229.

[41] M. Wu, J. Ye, A small sphere and large margin approach for novelty detection using training data with outliers, IEEE transactions on pattern analysis and machine intelligence 31 (11) (2009) 2088–2092.

[42] V. Mygdalis, A. Iosifidis, A. Tefas, I. Pitas, Semi-supervised subclass support vector data description for image and video classification, Neurocomputing 278 (2018) 51–61.

[43] B. Schölkopf, R. Herbrich, A. J. Smola, A generalized representer theorem, International Conference on Computational Learning Theory (2001) 416–426.

[44] P. Drineas, M. W. Mahoney, On the nyström method for approximating a gram matrix for improved kernel-based learning, journal of machine learning research 6 (Dec) (2005) 2153–2175.

[45] J. H. Metzen, T. Genewein, V. Fischer, B. Bischoff, On detecting adversarial perturbations, arXiv preprint arXiv:1702.04267.

[46] L. Ruff, N. Görnitz, L. Deecke, S. A. Siddiqui, R. Vandermeulen, A. Binder, E. Müller, M. Kloft, Deep one-class classification, in: International Conference on Machine Learning, 2018, pp. 4390–4399.

[47] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[48] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images, Tech. rep., Citeseer (2009).

[49] A. S. Georghiades, P. N. Belhumeur, D. J. Kriegman, From few to many: Illumination cone models for face recognition under variable lighting and pose, IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (6) (2001) 643–660.

[50] J. Stallkamp, M. Schlipsing, J. Salmen, C. Igel, The german traffic sign recognition benchmark: A multi-class classification competition., Proceedings of the International Joint Conference on Neural Networks (IJCNN) 6 (2011) 7.

[51] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, arXiv preprint arXiv:1801.04381.

[52] X. Wu, R. He, Z. Sun, T. Tan, A light cnn for deep face representation with noisy labels, IEEE Transactions on Information Forensics and Security 13 (11) (2018) 2884–2896.

[53] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[54] N. Carlini, D. Wagner, Adversarial examples are not easily detected: Bypassing ten detection methods, in: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, ACM, 2017, pp. 3–14.

[55] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.