MultiMagNet: A Non-deterministic Approach based on the Formation of Ensembles for Defending Against Adversarial Images

Gabriel R. Machado[1], Ronaldo R. Goldschmidt[1] and Eugênio Silva[2]
[1]*Section of Computer Engineering (SE/8), Military Institute of Engineering (IME), Rio de Janeiro, Brazil*
[2]*Computing Center (UComp), State University of West Zone (UEZO), Rio de Janeiro, Brazil*

Keywords: Artificial Intelligence and Decision Support Systems, Advanced Applications of Neural Networks.

Abstract: Deep Neural Networks have been increasingly used in decision support systems, mainly because they are the state-of-the-art algorithms for solving challenging tasks, such as image recognition and classification. However, recent studies have shown these learning models are vulnerable to adversarial attacks, *i.e.* attacks conducted with images maliciously modified by an algorithm to induce misclassification. Several works have proposed methods for defending against adversarial images, however these defenses have shown to be inefficient, since they have facilitated the understanding of their internal operation by attackers. Thus, this paper proposes a defense called MultiMagNet, which randomly incorporates at runtime multiple defense components, in an attempt to introduce an expanded form of non-deterministic behavior so as to hinder evasions by adversarial attacks. Experiments performed on MNIST and CIFAR-10 datasets prove that MultiMagNet can protect classification models from adversarial images generated by the main existing attacks algorithms.

1 INTRODUCTION

The advances that Deep Neural Networks have presented in recent years has been impulsed mainly by the increasingly necessity to analyse and interpret a massive and diversified amount of data (Obermeyer and Emanuel, 2016). Currently, information systems which make use of intelligent resources to support decisions in safety-critical environments involving computing vision tasks, such as (i) biometric recognition for users authentication (Derawi et al., 2010) (Bae et al., 2018), (ii) identification of handwritten digits or characters (Srivastava et al., 2019) (Tolosana et al., 2018) and (iii) surveillance systems (Ding et al., 2018), are among the various applications of Deep Neural Networks.

Nevertheless, recent work have demonstrated that the performance of Deep Neural Networks, which can even overcome the human perception (Karpathy, 2014), has significant drop in face of adversarial images (Szegedy et al., 2013) (Goodfellow et al., 2015) (Papernot et al., 2016a) (Carlini and Wagner, 2017c). Adversarial images contain malicious perturbations generated by an attack algorithm, usually minimal and imperceptible to human eyes, but can lead learning algorithms to misclassification. This attack is known as *adversarial attack* (Madry et al., 2017). The facility that attackers have into fooling classifiers can result in wide-range losses and accidents in real world scenarios, menacing the use of these learning models on several security-critical applications (Klarreich, 2016).

In order to minimize the harmful effects of the adversarial attacks, several defenses have been proposed. Many defense approaches consist of detecting the existence of possible perturbations in images before they misclassify learning models (Xu et al., 2018) (Gong et al., 2017) (Metzen et al., 2017) (Hendrycks and Gimpel, 2017). However, works such as (Carlini and Wagner, 2017a) and (He et al., 2017) have demonstrated that these defenses have limitations in terms of (i) their ability to detect adversarial images generated by diferent attacks algorithms and/or (ii) their utilization of deterministic approaches, *i.e.* defenses that, at each execution, always uses the same procedure, which facilitates the attacker's understading of the defense's *modus operandi*.

As a means of reducing the dependence of the defense for specific attack algorithms and hindering the prediction of its behaviour by the attacker, a recent work has developed a defense, called MagNet (Meng and Chen, 2017). MagNet is a non-deterministic defense which randomly chooses at runtime a compo-

Machado, G., Goldschmidt, R. and Silva, E.
MultiMagNet: A Non-deterministic Approach based on the Formation of Ensembles for Defending Against Adversarial Images.
DOI: 10.5220/0007714203070318
In *Proceedings of the 21st International Conference on Enterprise Information Systems (ICEIS 2019)*, pages 307-318
ISBN: 978-989-758-372-8

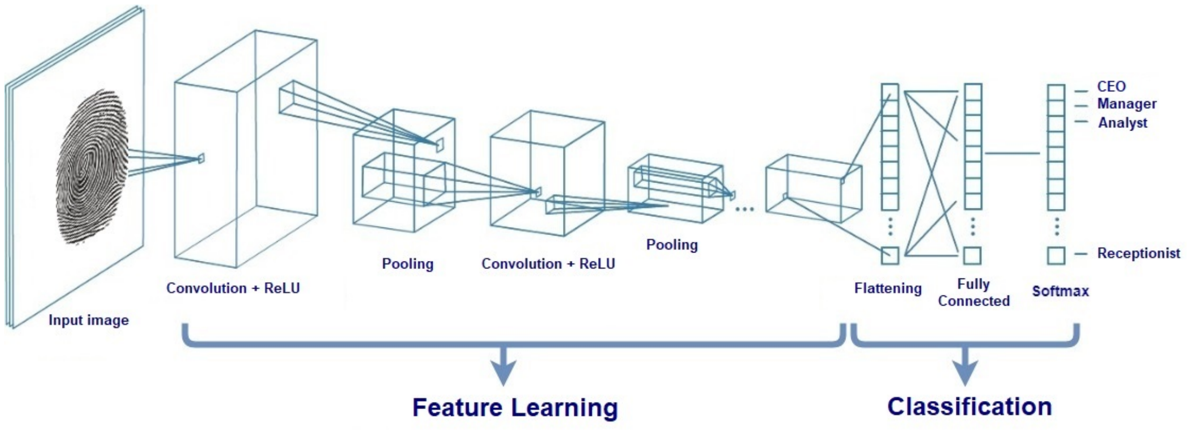

Figure 1: An example of a CNN containing several convolutional layers.[1]

nent, used to compute a threshold needed to classify the input images as legitimate or adversarial. Afterwards, the images classified as legitimate are reconstructed by another component before being sent to the classifier. Despite presenting some promising results before various adversarial attacks, (Carlini and Wagner, 2017b) have shown that, even with the non-deterministic effect provided by the random selection of a defense component, MagNet can be evaded by adversarial images. Thus, this work raises the hypothesis that the selection of multiple defense components can amplify the non-deterministic effect, reducing the predictability of the defense method's behaviour, becoming it more robust than MagNet against different adversarial attacks algorithms.

Therefore, this paper proposes MultiMagNet, an adversarial image detection method which arranges ensembles at runtime by randomly choosing multiple defense components, implemented as autoencoders. Experiments performed on MNIST (LeCun et al., 1998) and CIFAR-10 datasets (Krizhevsky and Hinton, 2009) indicate the veracity of the hypothesis by showing that MultiMagNet has presented better results than MagNet in the majority of attack scenarios evaluated.

This paper is structured as follows: Section 2 brings the needed background to understand the work. Section 3 summarizes the main defense methods against adversarial images available in literature. Section 4 details the proposed defense method. Section 5 describes the experiments performed and discusses the obtained results. Finally, Section 6 brings the final considerations, highlights the main contributions of this work and indicates suggestions for future works.

2 BACKGROUND

2.1 Convolutional Neural Networks

The Convolutional Neural Networks (CNNs), a special type of Deep Neural Network, are the state-of-the-art learning models in image classification and recognition tasks (He et al., 2016) (Hu et al., 2017) and, for this reason, they have become the prime target of adversarial attacks. The CNNs, unlike the conventional neural networks, are able to learn automatically the main features of a image by reducing its representation space. After the extraction of the most important features, the fully-connected layer acts in a way similar to a regular neural network, with the difference of producing as an output the probabilities of the input image belonging to each class of the problem being studied. These probabilities are computed using the softmax function at the last layer of the neural network. Figure 1 shows an example of a CNN architecture. More details about the CNNs can be obtained at (Goodfellow et al., 2016).

2.2 Autoencoders

Autoencoders are neural networks trained to reconstruct an input x, generating as an output an approximation x', with the smallest reconstruction error as possible (Goodfellow et al., 2016). Formally, an autoencoder $ae = d \circ e$ comprises two components: (i) an encoder $e : \mathbb{S} \to \mathbb{H}$ and a decoder $d : \mathbb{H} \to \hat{\mathbb{S}}$, where $\mathbb{S}$ is the input space, $\mathbb{H}$ is the compressed space learnt by the encoder component and $\hat{\mathbb{S}}$ represents the input

[1]Adapted from https://www.mathworks.com/discovery/convolutional-neural-network.html. Accessed in June 23, 2018.

space reconstructed by the autoencoder. The reconstruction error $ER_{ae(x)}$ is the difference between the input x and its reconstructed version $x' = ae(x)$ as defined by Equation 1, where p is the distance metric:

$$ER_{ae(x)} = ||x - ae(x)||_p \quad (1)$$

Autoencoders usually are used for (i) dimensionality reduction and (ii) feature learning, once they preserve only the most important features of the data (Goodfellow et al., 2016).

2.3 Adversarial Images

An adversarial image is a image which contains a minimal perturbation[2], oftentimes imperceptible to human eyes, generated by a malicious algorithm in order to induce learning models to misclassification (see Figure 2). Formally, given a model F trained using legitimate images, x an input image such that $x \in \mathbb{R}^{w \times h \times c}$, where w and h are the dimensions of image x and c its quantity of color channels, it is generated an image x_{adv}, such that $x_{adv} = x + \delta x$, where δx is the perturbation and, in a succeeded adversarial attack, $F(x) \neq F(x')$.

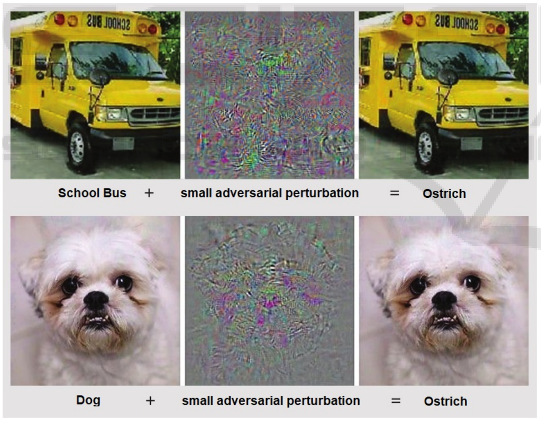
School Bus
+
small adversarial perturbation
=
Ostrich
Dog
+
small adversarial perturbation
=
Ostrich

Figure 2: Malicious and usually imperceptible perturbations in a input image can induce trained models to misclassification. Adapted from (Klarreich, 2016).

2.4 Adversarial Attacks

Adversarial attacks are malicious optimization algorithms which generate and insert perturbations into legitimate images in order to lead previously trained models to misclassification. There are several algorithms available in literature, however, for the experiments performed in this work, it has been used the four most used adversarial attacks: (i) FGSM, (ii) BIM, (iii) DeepFool e (iv) CW. These adversarial attacks are explained in the following.

Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015): FGSM is a non-iterative attack algorithm, whose main characteristic is its linear complexity. The linear complexity of FGSM is computationally efficient, however it contributes to generate larger perturbations than the ones generated by iterative algorithms. Given an image $x \in \mathbb{R}^{w \times h \times c}$, FGSM generates an adversarial image x_{adv} using Equation 2.

$$x_{adv} = x - \varepsilon \cdot sign(\nabla_x J(\Theta, x, y)) \quad (2)$$

In Equation 2, Θ represents the network parameters, y the respective class of x, ε the maximum perturbation which can be inserted into the image x and $J(\Theta, x, y)$ is the cost function used to train the network.

Basic Iterative Method (BIM) (Kurakin et al., 2016a): BIM is the iterative version of FGSM. Unlike FGSM that executes only a step of size ε towards the gradient descent, BIM executes several smaller steps α, where the result is upper bounded by ε in order to prevent the amount of perturbation does not exceed the quantity desired by the attacker. Formally, BIM is a recursive method that computes x_{adv} according to Equation 3:

$$x_{adv} = \begin{cases} x_0^{adv} = 0 \\ x_i^{adv} = x_{i-1}^{adv} - clip(\alpha \cdot sign \nabla_x J(\Theta, x_{i-1}^{adv}, y)) \end{cases} \quad (3)$$

DeepFool (Moosavi-Dezfooli et al., 2016): The ideia behind DeepFool consists of finding the closest decision boundary from a legitimate image x in the image space. Afterwards, x is subtly perturbated so as to make it cross the decision boundary and fool the classifier. Due to the high dimensionality of the image, DeepFool adopts an iterative approach of linear approximation. In each iteration, DeepFool linearizes the model around the intermediate x_{adv} and computes an optimal update direction in the linearized model. Then, x_{adv} is updated on this direction by a small step α.

Carlini & Wagner Attack (CW) (Carlini and Wagner, 2017c): The CW attack is the state-of-the-art algorithm to generate adversarial images. Formally, CW is an iterative attack where, given a CNN F with Z as the penultimate layer, called *logits*, and an legitimate image x belonging to the class t, CW uses the gradient descent to solve Equation 4:

$$\text{minimize } ||x - x_{adv}||_2^2 + c \cdot \ell(x_{adv}) \quad (4)$$

where the cost function $\ell(x_{adv})$ is defined in Equation 5.

[2]A perturbation is a systematic distortion maliciously generated in an image by an attack algorithm.

$$\ell(x_{adv}) = max(max\{Z(x_{adv})_i : i \neq t\} - Z(x_{adv})_t, -conf) \quad (5)$$

In Equation 5, the hyperparameter $conf$ indicates the confidence level of the CW attack. Higher values of $conf$ usually produce adversarial images with greater ability to fool classifiers, however, these images will also contain more perturbations within.

2.5 Jensen-Shannon Divergence

The Jensen-Shannon Divergence (JSD) computes the divergence between two probabilistic distributions P and Q. In this work, the distributions P and Q are obtained by the output of *softmax* layer F of a CNN where, given two images x and y, $P = F(x)$ and $Q = F(y)$. $P(i)$ e $Q(i)$ indicate respectively the probabilities of the images x and y belong to the class i. The value of the JSD metric, given two images x and y, is computed by Equation 6.

$$JSD(P||Q) = \frac{1}{2}D_{KL}(P||M) + \frac{1}{2}D_{KL}(Q||M) \quad (6)$$

where

$$M = \frac{1}{2}(P+Q), D_{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

3 RELATED WORK

The defense of classification models against adversarial attacks is not a trivial task and several strategies have already been proposed in literature. The most relevent work in this subject and their respective approaches to defend against adversarial attacks are next discussed.

Adversarial Training (Szegedy et al., 2013), (Goodfellow et al., 2015), (Kurakin et al., 2016b), (Madry et al., 2017), (Kannan et al., 2018): The adversarial training, also known as robust optimization, is a proactive defense approach[3] based on data augmentation[4] to train classifiers in a dataset containing legitimate and adversarial images, thus forcing the classification model to produce correct outputs to the malicious images. This strategy has two important limitations: (i) it is computationally expansive and (ii) it is deterministic, once it creates dependencies between the detection method and the attack algorithms used in the adversarial training process.

Defensive Distillation (Papernot et al., 2016b): is a proactive and deterministic defense which trains a model F in a dataset X containing legitimate samples and labels Y, generating as output the probabilities set $F(X)$. Afterwards, the original labels set Y are then replaced by the $F(X)$ probabilistic set, and a new model F_d with the same architecture of F is created and trained with dataset X yet using the probabilistic labels $F(X)$. After training, the obtained model F_d, called distilled model, produces the probabilistic distilled outputs F_d. Classifiers using defensive distillation are based on *gradient masking*: an effect that hides the classifier's gradient in order to hinder the generation of adversarial images by the attacker (Papernot et al., 2017). Defenses based on gradient masking can be easily bypassed since the attacker can create his own classifier, generate adversarial images using this classifier and a more elaborated attack algorithm, and finally transfer these recently generated images to the distilled classifier (Carlini and Wagner, 2017c).

Feature Squeezing (Xu et al., 2018): reactive defense which is based on the hypothesis that the high dimensional image spaces facilitates attackers into generating stronger perturbations. Therefore, the authors basically have used two methods for reducing the dimensionality of the images, in order to remove possible perturbations that may be present in them: (i) color bit depth reduction and (ii) spatial smoothing. Using a classifier and a predefined threshold, a comparison is performed among the prediction of the original image x with the predictions of its versions x', reduced by the method (i), and x'', reduced by the method (ii), respectivelly. If one of these comparisons is above the threshold, x is labeled as adversarial and disposed before reaching the classifier. Despite presenting good results before CW attack, (He et al., 2017) has shown it is possible to evade it, mainly because its deterministic architecture, which always chooses the same dimensionality reduction techniques at each execution.

MagNet (Meng and Chen, 2017): MagNet is a reactive and non-deterministic method that is formed by two defense layers: (i) the detection layer which rejects, using a predefined threshold, the images far from the decision boundary, once they contain more perturbations, and (ii) a reformer layer that receives the images coming from the detection layer and reconstructs them in order to remove any undetected perturbations. After the reconstruction, the image is sent to the classifier. MagNet chooses randomly two

[3]Defenses against adversarial images are divided into (i) proactive defenses, which aim to make models more robust in classifying adversarial images and (ii) reactive defenses, which act as detectors of adversarial images, preventing them from reaching the classifier.

[4]Procedure performed in a dataset in order to increase the amount of samples used to train classification models.

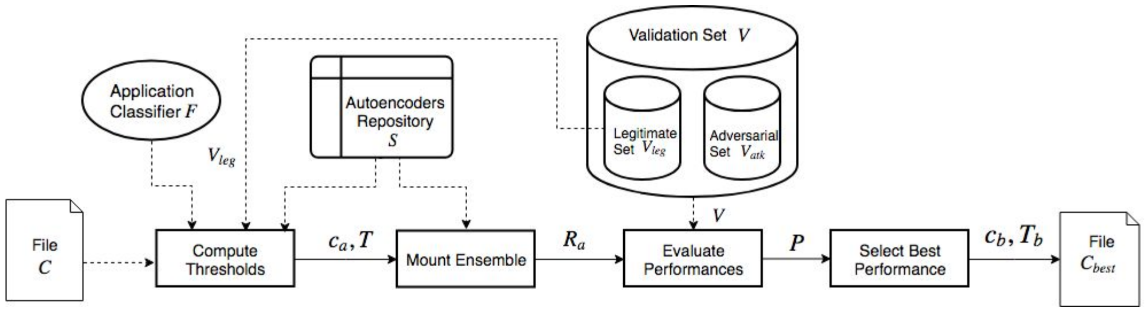

Figure 3: Schematic flowchart of the Calibration Stage.

autoencoders from a repository: one for the detection layer and the other one for the reformer layer. Despite the use of randomness for choosing both autoencoders, (Carlini and Wagner, 2017b) has shown that MagNet can be evaded by adversarial images.

4 PROPOSED DEFENSE

The defense method proposed in this paper, called MultiMagNet, is an extension of MagNet. The MultiMagNet's operation basically comprises two stages: (i) the *Calibration Stage* and (ii) the *Deployment Stage*. All the processes which compose the *Calibration Stage* are going to be explained first. Later, the processes related to the *Deployment Stage* will also be discussed.

4.1 The Calibration Stage

In the *Calibration Stage* (see Figure 3), all the MultiMagNet's desired combinations of hyperparameters, predefined by the user as inputs in a file C, are once evaluated using the validation set V and the application classifier F. At the end of the *Calibration Stage*, the best combination of hyperparameters $c_b \in C$ and the thresholds set T_b (computed using the hyperparameters c_b) are then saved in a file called C_{best}. The file C_{best}, once defined, will prompty provide the best tuple of hyperparameters c_b and the thresholds T_b for the *Deployment Stage*, avoiding unnecessary and repetitive computation. All the processes related to the *Calibration Stage* will be next described in details.

4.1.1 Compute Thresholds

The first process, *Compute Thresholds*, receives as input a file C, defined beforehand by the user, which contains his desired set of values to be evaluated for the following MultiMagNet's hyperparameters: (i) the false positive rate set T_{fp}, such that $t_{fp_a} \in T_{fp}$ and $0 < t_{fp_a} < 1$, (ii) the temperature set K, where $k_a \in K, k_a \geq 1$, (iii) the metric set M, which $me_a \in M$ can be the Reconstruction Error (RE), defined by Equation 1, or the Jensen-Shannon Divergence (JSD), defined by Equation 6 (formally, $me_a \in \{RE, JSD\}$), and (iv) the threshold approach set $\mathcal{T}$, where $\mathcal{T} = \{minTA, MTA\}$, $\delta_a \in \mathcal{T}$. These hyperparameters are going to be explained in Section 4.1.3. Besides the file C, this process also receives as inputs the validation subset V_{leg} containing only legitimate images, the respository S containing m autoencoders[5] and the application classifier F. This process gives as output the set T, which contains $m \times c$ computed thresholds, where m is the number of autoencoders in repository S and c is the number of possible combinations of user-predefined values in the file C to be evaluated for the hyperparameter sets T_{fp}, K, M and $\mathcal{T}$, such that $c = |T_{fp}| \times |K| \times |M| \times |\mathcal{T}|$ and $a \leq c$.

To compute all the $m \times c$ thresholds of T, the legitimate images from the validation set V_{leg} were initially reconstructed for each autoencoder $s_i \in S, i \leq m$, forming the V_{L_i} set, where $V_{L_i} = \{vl_i | vl_i = s_i(v_l), v_l \in V_{leg}\}$, and $s_i(v_l)$ represents the image v_l reconstructed by the autoencoder s_i. Next, the classification thresholds are defined by RE or JSD metrics, according to the current hyperparameter δ_a.

Regarding the metric RE, it has been used Equation 1 to compute the reconstruction errors of each legitimate image v_l and its reconstructed version vl_i (where, in Equation 1, $x = v_l$, $ae(x) = vl_i$ and $p = 1$), thus forming the RE_i array. After computing all the reconstruction errors in V_{leg} set using the autoencoder s_i, the values in RE_i array are then sorted in descending order and the hyperparameter $t_{fp_a} \in T_{fp}$ is applied[6], giving as result the threshold $\tau_i = RE_i[t]$, where t is the corresponding index, such that $t = t_{fp_a} \times |V_{leg}|$.

On the other hand, for the metric JSD, it has been

[5]All of the m autoencoders in S are previously trained in the training dataset Tr, which contains only legitimate images.

[6]The false positive rate t_{fp_a} represents the percentage of legitimate images which can be misclassified as adversarial.

necessary to obtain the probabilistic softmax outputs from the last layer of the application classifier (represented by a CNN) for each legitimate image from V_{leg} and its respective reconstruted version. It is worth mentioning that the softmax outputs are related to the classes of the original classification problem. The Equation 7 describes the softmax function used.

$$F(l_i) = \frac{\exp(l_i/k_a)}{\sum_{j=1}^{n} \exp(l_j/k_a)} \tag{7}$$

In Equation 7, F represents the softmax layer of the application classifier, l, also known as *logits*, represents the output from the penultimate layer of the application classifier, i represents the corresponding index of the class with respect to the original classification problem and n means the total number of classes relating to the original classification problem. The hyperparameter $k_a \in K$ (where $k_a \geq 1$) is used to normalize the values of l_i e l_j, in order to prevent saturation (Meng and Chen, 2017). After computing the softmax outputs corresponding to the legitimate images from V_{leg} subset and their reconstructed versions V_{L_i}, it is calculated the Jensen-Shannon Divergence (JSD) from Equation 6, where $P = F(v_l)$, $Q = F(vl_i)$. According to (Meng and Chen, 2017), the utilization of the JSD metric is based on the hypothesis that the divergence of the softmax outputs between an legitimate image x_{leg} and its reconstruct version xr_{leg} is usually smaller than the divergence between the softmax outputs of an adversarial image x_{adv} and its reconstructed version xr_{adv}, such that $JSD(F(x_{leg}), F(xr_{leg})) < JSD(F(x_{adv}), F(xr_{adv}))$. In a way similar to the RE metric, all the divergences computed using the legitimate validation set V_{leg} and an autoencoder s_i are kept in an array JSD_i, which is also sorted in descending order and $t_{fp_a} \in T_{fp}$ is applied, giving as result the threshold $\tau_i = JSD_i[t]$, where $t = t_{fp_a} \times |V_{leg}|$.

Thus, by using the hyperparameters $k_a \in K$, $me_a \in \{RE, JSD\}$ and $t_{fp_a} \in T_{fp}$, all predefined by the user in file C, it is produced as output the T set containing $m \times c$ thresholds.

4.1.2 Mount Ensemble

The *Mount Ensemble* process receives as inputs the repository S containing m different autoencoders, the T set containing $m \times c$ thresholds and the current combination c_a to be tested in V set, where $c_a \in C$ and $c_a = (t_{fp_a}, k_a, me_a, \delta_a)$, $a \leq c$. It gives as output an ensemble R_a, which is formed as described as follows: n autoencoders are randomly chosen from S, where $n < m$ and $n \bmod 2 = 1$ (to ensure no ties in the vote count). After choosing the n autoencoders, their respective thresholds are also loaded from T set, according to the current combination c_a, thus forming the R_a set. Formally, $R_a = \{r_{1a}, r_{2a}, \cdots, r_{na}\}$, where r_{ia} is the pairwise (r_i, τ_{ia}), such that $r_i \in S$, $i \leq n$ and $\tau_{ia} \in T$, $a \leq c$.

4.1.3 Evaluate Performance

The *Evaluate Performance* process receives two inputs: (i) the ensemble R_a and (ii) the validation set $V = V_{leg} \cup V_{atk}$. V_{atk} is formed by 2,000 adversarial images generated from the 2,000 legitimate images in V_{leg} set using an attack algorithm atk, where $atk \in \{FGSM, BIM, DeepFool, CW\}$.

Each one of the 4,000 images in V is therefore reconstructed by all the autoencoders in R_a set, forming the set VR_a. Afterwards, it is used Equation 1 or 6, according to the current metric me_a, to compute the metric value set M_{V_a}. If $me_a = RE$, it is applied the Equation 1, where $x = v_i$, $ae(x) = vr_i$ and $p = 1$, such that $v_i \in V$, $vr_i \in VR_a$. If $me_a = JSD$, it is applied the Equation 6, where $P = F(v_i)$ and $Q = F(vr_i), v_i \in V, vr_i \in VR_a$.

Finally, each metric value $m_i \in M_{V_a}$ is compared to the threshold τ_a, which is computed based on the current value approach of δ_a, which can be: (i) **minimum threshold** (*minTA*) or (ii) **multiple threshold** (*MTA*). When the *minTA* approach is selected, it is considered the smallest threshold among all the associated thresholds for the n autoencoders in R_a, *i.e.* $\tau_a = min\{\tau_{1a}, \tau_{2a}, \cdots, \tau_{na}\}$. On the other hand, when is selected the *MTA* approach, it is considered each associated threshold for all the autoencoders in R_a, *i.e.* $\tau_a \in \{\tau_{1a}, \tau_{2a}, \cdots, \tau_{na}\}$. When $m_i \leq \tau_a$, a variable q_{leg} (which represents the votes of v_i being legitimate) counts a vote, otherwise q_{adv} counts a vote. By majority vote, the image is classified as legitimate if $q_{leg} > q_{atk}$, otherwise is classified as adversarial. At the end of the vote count for each image v_i, it is produced a confusion matrix Ma as defined in Equation 8.

$$Ma = \begin{bmatrix} TN & FN \\ FP & TP \end{bmatrix} \tag{8}$$

In Equation 8, the elements TN, FN, FP and TP in matrix Ma define respectively the amount of (i) adversarial images voted as adversarial, (ii) adversarial images voted as legitimate, (iii) legitimate images voted as adversarial and (iv) legitimate images voted as legitimate. After computing the c confusion matrices, it is finally produced as output by the Evaluate Performance process the P set, formed by c tuples, where each tuple contains the following three metrics: (i) accuracy (ACC), (ii) positive predictive value

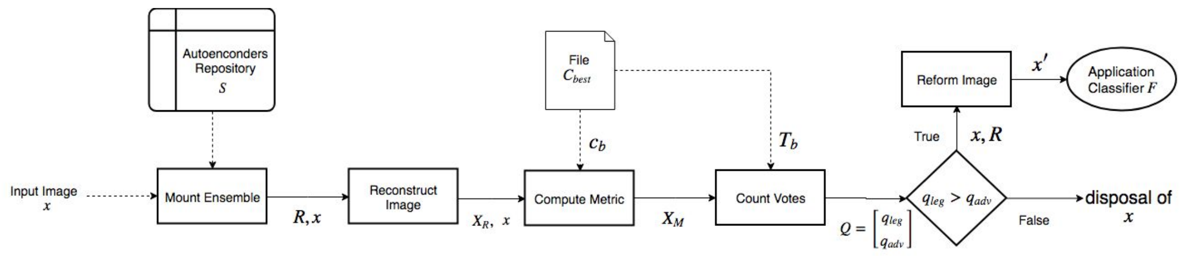

Figure 4: Schematic flowchart of the Deployment Stage.

(PPV) and (iii) negative predictive value (NPV). All these metrics are formally defined in Equation 9.

$$
\begin{aligned}
ACC &= {}^{TN+TP}/_{TN+FN+FP+TP} \\
PPV &= {}^{TN}/_{TN+FN} \\
NPV &= {}^{TP}/_{TP+FP}
\end{aligned}
\tag{9}
$$

4.1.4 Select Best Performance

The *Select Best Performance* process receives as input the P set containing c performance tuples. It is produced as output the file C_{best} which contains (i) the combination of hyperparameters c_b that leads to the MultiMagNet's highest accuracy, where $c_b = (t_{fp_b}, k_b, me_b, \delta_b)$ and (ii) the T_b set. The T_b set, such that $T_b \subset T$, contains the n thresholds computed by using the c_b combination of hyperparameters, such that $c_b \in C$. The highest accuracy metric on V set has been chosen to elect the best set of hyperparameters c_b, since it can describe a more general performance scenario.

4.2 Deployment Stage

Once calibrated, MultiMagNet is ready to receive input images in the *Deployment Stage*. After forming an ensemble R and loading its best set of hyperparameters c_b and the respective thresholds T_b from file C_{best}, MultiMagNet's ensemble R can return a verdict whether an input image x is legitimate or not. The classification made by MultiMagNet is performed by majority vote, where all the n votes, representing each autoencoder in R, are counted. In case of being classified as legitimate, the image x is reconstructed by an autoencoder randomly selected from R, in order to remove any undetected perturbations, and afterwards x is finally sent to the application classifier F. If x is classified as adversarial, MultiMagNet simply discards it before reaching F. Figure 4 illustrates the five processes belonging to the Deployment Stage, and each of them will be explained in the following.

4.2.1 Mount Ensemble

This process receives two inputs: (i) the repository S containing m autoencoders and (ii) the input image x to be evaluated by MultiMagNet. It gives as output the ensemble R containing n autoencoders chosen at random, where $n \leq m, n \bmod 2 = 1$.

4.2.2 Reconstruct Image

The *Reconstruct Image* process receives two inputs: (i) the input image x and (ii) the ensemble R. It gives as outputs the image x itself and the X_R set, which is formed by the reconstructed versions xr_i of the image x made by each autoencoder $r_i \in R$.

4.2.3 Compute Metric

The *Compute Metric* process receives three inputs: (i) the input image x, (ii) the X_R set containing the n reconstructed versions of x, such that $xr_i \in X_R$, and the file C_{best}, which contains the tuple c_b of hyperparameters that has led to the best accuracy in *Calibration Stage*. In a way similar to the procedures explained in Sections 4.1.1 and 4.1.3 for computing the metric values, this process returns the X_M set, which contains the metric values among x and its reconstructed versions in X_R set, which can be computed from Equation 6 if $me_b = JSD$, where $P = F(x)$ e $Q = F(xr_i)$, or from Equation 1, where $ae(x) = xr_i$ and $p = 1$.

4.2.4 Count Votes

The *Count Votes* process also works in a very similar way when compared to the vote count performed by *Evaluate Performance* process in *Calibration Stage* (see Section 4.1.3). It receives as inputs the X_M set and the T_b set provided by C_{best} file. T_b contains the n thresholds that have been defined in *Calibration Stage*. It returns q_{leg} and q_{adv}, which respectively represent the count of votes for x being legitimate or adversarial. If $q_{leg} < q_{adv}$, the input image x is discarted, otherwise x is sent to the last process, *Reform Image*.

Table 1: Hyperparameters defined for each attack algorithm.

Attack	Datasets and Parameters
FGSM	MNIST: $\varepsilon = 0.2$ CIFAR-10: $\varepsilon = 0.025$
BIM	MNIST: $\varepsilon = 0.15$; $\alpha = 0.07$; 50 iterations CIFAR-10: $\varepsilon = 0.025$; $\alpha = 0.01$; 1,000 iterations
DeepFool	MNIST: overshoot = 0.02; max iter = 50 CIFAR-10: overshoot = 0.02; max iter = 50
CW	MNIST: $conf$ = 0.0; binary searches = 1; lrate = 0.2; initial const = 10, max iter = 100 CIFAR-10: $conf$ = 0.0; binary searches = 1; lrate = 0.5; initial const = 1, max iter = 100

4.2.5 Reform Image

The *Reform Image* process receives as inputs: (i) the image x classified as legitimate by the ensemble of autoencoders R and (ii) the ensemble R itself. After receiving the image x, it is chosen randomly from R an autoencoder r_i which reconstructs the image x as an attempt to remove any remaining perturbations from it, thus producing an resulting image x', such that $x' = r_i(x)$. Afterwards, the reconstructed image x' is finally sent to be classified by F.

5 EXPERIMENTAL SETUP AND RESULTS

5.1 Datasets

In order to perform the experiments, it was chosen MNIST (LeCun et al., 1998) and CIFAR-10 (Krizhevsky and Hinton, 2009) datasets, since they are widely used by several related works (Xu et al., 2018), (Meng and Chen, 2017), (Zantedeschi et al., 2017), (Carlini and Wagner, 2017c). The MNIST dataset contains 60,000 greyscale images of handwritten digits distributed in 10 different classes, with dimensions of $28 \times 28 \times 1$, which respectively represent 28 pixels of width, 28 pixels of height and 1 color channel. The CIFAR-10 dataset, on the other hand, contains 60,000 colorful images, with dimensions $32 \times 32 \times 3$, distributed in 10 different classes. In the experiments performed on each dataset, their respective images have been partitioned as follows: the first 45,000 images have formed the Tr set, in order to train the m autoencoders in repository S and the application classifier F. The following 5,000 images have formed the legitimate validation set V_{leg}, and the remaining 10,000 images have formed the Te set, destinated to test and evaluate the autoencoders, the application classifier, the MagNet (simulated when it is chosen only one autoencoder to form the ensemble R) and the MultiMagNet. All the images have been normalized to have their pixels' intensity values in the interval $[0,1]$, instead of the original interval of $[0,255]$.

To generate adversarial images, it has been used four different attack algorithms: FGSM, BIM, DeepFool and CW. However, due to the high computational cost to generate adversarial images, it has been necessary to define a smaller test set called D, according to the following criterion: from the 10,000 legitimate images in Te, it has been randomly selected 2,000 images. These selected images have been labeled as *legitimate* and kept in D_{leg}. Next, each one of the four attack algorithms have been applied to the images in D_{leg}, and the resulting images kept in D_{FGSM}, D_{BIM}, $D_{DeepFool}$ and D_{CW}, respectively[7]. Table 1 shows the hyperparameters empirically defined in *Calibration Stage* for each attack algorithm. At last, the D set has been formed by the union of the sets D_{leg} and D_{atk}, *i.e.* $D = D_{leg} \cup D_{atk}$, where $atk \in \{FGSM, BIM, DeepFool, CW\}$.

5.2 Autoencoders and Classifiers

After training, the application classifier F adopted for MNIST dataset has presented an accuracy of 99.40% on Te set. Likewise, the application classifier adopted for CIFAR-10 (a CNN with architecture All Convolutional Net (Springenberg et al., 2014)) has presented an accuracy of 89.76%. Both results are good approximations of the state-of-the-art. The repository S of MultiMagNet has been set with 10 autoencoders formed by different architectures and parameters, previously trained on Tr set. After training, all the autoencoders have presented reconstruction errors on Tr set below 10^{-3}.

[7]In order to generate adversarial images, it has been used the framework *Adversarial Robustness Toolbox (ART)* (Nicolae et al., 2018), which contains the implementations for all the four aforementioned attack algorithms.

Table 2: MagNet and MultiMagNet's performance on MNIST dataset for 40 experiments.

Defense	Attack	ACC (μ - σ)	PPV (μ - σ)	NPV (μ - σ)
MagNet	FGSM	99.90% - 0.37%	99.81% - 0.70%	100.00% - 0.00%
	BIM	99.90% - 0.30%	99.81% - 0.58%	100.00% - 0.00%
	DeepFool	96.43% - 2.20%	92.83% - 4.30%	100.00% - 0.00%
	CW	56.00% - 5.17%	94.84% - 2.99%	27.54% - 7.85%
MultiMagNet	FGSM	99.92% - 0.26%	99.85% - 0.51%	100.00% - 0.00%
	BIM	99.92% - 0.26%	99.85% - 0.54%	100.00% - 0.00%
	DeepFool	96.47% - 1.48%	92.90% - 3.08%	100.00% - 0.00%
	CW	61.10% - 5.74%	45.99% - 22.23%	75.24% - 15.25%

Table 3: MagNet and MultiMagNet's performance on CIFAR-10 dataset for 40 experiments.

Defense	Attack	ACC (μ - σ)	PPV (μ - σ)	NPV (μ - σ)
MagNet	FGSM	58.43% - 5.56%	90.15% - 4.07%	26.35% - 8.51%
	BIM	77.72% - 3.91%	88.98% - 3.50%	69.75% - 6.42%
	DeepFool	71.88% - 21.18%	93.03% - 2.73%	49.93% - 42.93%
	CW	66.55% - 6.53%	98.43% - 4.40%	45.20% - 10.95%
MultiMagNet	FGSM	65.88% - 4.48%	67.13% - 12.64%	65.06% - 13.22%
	BIM	79.38% - 6.13%	66.57% - 15.42%	89.11% - 6.95%
	DeepFool	90.25% - 4.66%	80.67% - 9.22%	99.86% - 0.69%
	CW	68.95% - 4.45%	48.82% - 8.80%	84.31% - 5.96%

5.3 Prototype

The MultiMagNet prototype has been developed in Python, by using the following frameworks: (i) *Tensorflow*, (ii) *Keras*, (iii) *SciKit-Learn*, (iv) *NumPy* and (v) *Scipy*. All the source code, including the architectures and parameters defined for the autoencoders and applications classifiers are available for consultation and/or download [8]. The adversarial images used in the experiments are also available for download [9]. All the experiments have been conducted on a single machine with the following setup: CPU i7 3770, 16GB RAM and a GPU GTX 1060 with 1280 CUDA cores.

5.4 Results

It has been performed 40 experiments on the respective D sets of MNIST and CIFAR-10 datasets. For every 100 input images (once D set has a total of 4,000 images), a new ensemble R containing n different autoencoders has been formed, where $n \in \{3,5,7,9\}$ for MultiMagNet and $n = 1$ for MagNet. It is worth remembering that n must be an odd number to avoid ties. Tables 2 and 3 show the results of the mean μ and standard deviation σ obtained by MagNet and MultiMagNet (based on 40 experiments) using the metrics defined by Equation 9.

When analyzing the results of MagNet and MultiMagNet included in Tables 2 and 3, it becomes clearer that MultiMagNet has presented better performance on the metric *NPV* than MagNet for all attack algorithms on both datasets. The good performance presented by MultiMagNet on *NPV* metric indicates its better ability to detect adversarial images, which is priority in most security-critical applications. However, by analyzing Table 4, it can be noticed that the adoption of the minimum threshold approach (*minTA*) in most of attack scenarios may explain the increase of the *NPV* metric, mainly because the *minTA* metric assigns the smallest computed threshold for all the n chosen autoencoders. In addition to the minimum threshold approach, the false positive rate hyperparameter t_{fp} may have also influenced in the fall of *PPV* metric, since smaller values of t_{fp} than the ones in Table 4 have produced an increase in the *PPV* metric to the detriment of the *NPV* metric. Nevertheless, the fall presented by MultiMagNet in the *PPV* metric is justifiable, due to the fact that it has obtained greater accuracy than MagNet on both datasets, before all the four evaluated attacks, reaching differences up to 18.37 percentage points. Such comparison points to the validation of the hypothesis defended by this work.

Regarding the Table 6, which shows how strong each attack was on leading F to misclassification, it is worth mentioning that, although the CW attack is the state-of-the-art in generating adversarial images, on MNIST dataset it has not been the attack algo-

[8]https://github.com/gabrielrmachado/MultiMagNet

[9]https://drive.google.com/open?id=1l5KHwpbWLLgcv34AGUF3Fq3z3fuXV22z

Table 4: MultiMagNet's hyperparameters defined for each attack algorithm.

Dataset	Attack	Drop Rate	Threshold Approach	Metric	K
MNIST	FGSM - BIM	0.001	*MTA*	RE	-
	DeepFool	0.07	*MTA*	JSD	1
	CW	0.05	*minMTA*	RE	-
CIFAR-10	FGSM - BIM	0.1	*minMTA*	JSD	15
	DeepFool	0.07	*minMTA*	JSD	1
	CW	0.07	*minMTA*	JSD	5

Table 5: Accuracies obtained by the application classifier F on D set when evaluated in five different scenarios.

Dataset	Attack	No defense (μ - σ) (%)	MagNet only (μ - σ) (%)	MultiMagNet only (μ - σ) (%)	MagNet and Reformer (μ - σ) (%)	MultiMagNet and Reformer (μ - σ) (%)
MNIST	FGSM	79.53 - 4.52	99.37 - 1.11	99.46 - 1.06	98.98 - 1.75	99.17 - 1.41
	BIM	81.45 - 3.29	99.40 - 1.03	99.49 - 1.20	99.16 - 1.18	99.19 - 1.61
	DeepFool	50.00 - 5.38	99.95 - 0.31	99.95 - 0.31	99.90 - 0.46	99.95 - 0.31
	CW	81.60 - 4.33	92.92 - 3.21	98.71 - 1.72	96.19 - 2.20	98.89 - 1.36
CIFAR-10	FGSM	64.83 - 2.94	67.27 - 1.01	72.81 - 3.59	70.08 - 2.49	72.91 - 2.75
	BIM	69.30 - 1.85	89.12 - 2.92	89.94 - 1.42	92.58 - 2.41	96.75 - 3.54
	DeepFool	65.97 - 2.03	79.12 - 10.05	92.52 - 1.67	87.15 - 4.48	92.91 - 2.88
	CW	47.03 - 1.29	59.14 - 3.29	71.86 - 5.74	60.08 - 3.27	72.36 - 5.22

rithm which fooled at most the application classifier F (when compared to DeepFool attack). This is mainly due to two reasons: (i) CW is the attack which contains the largest number of hyperparameters (see Table 1); (ii) the MNIST images contain much less information on them (when compared to the CIFAR-10 images), what have proved to be more computationally expansive to find a set of hyperparameters for the CW attack which could produce more harmful images to F without increasing their amount of perturbation. Nonetheless, according to Table 2, the adversarial images produced by CW attack on MNIST dataset have been the most difficult ones to be detected.

Table 6: Accuracies obtained by the application classifier F on D_{atk} set, without any previous defense.

Dataset	Attack	Accuracy D_{atk} set (no defense)
MNIST	FGSM	59.65%
	BIM	63.50%
	DeepFool	0.60%
	CW	63.80%
CIFAR	FGSM	40.60%
	BIM	49.55%
	DeepFool	42.70%
	CW	5.00%

Finally, in addition to the scenario depicted by Table 6, the application classifier F has been evaluated in more five scenarios: (i) performance on D set without any defense, (ii) performance on D set with MagNet, (iii) performance on D set with MultiMagNet, (iv) performance on D set with MagNet and Reformer[10] and (v) performance on D set with MultiMagNet and Reformer. It is important to highlight that it has been also computed the mean μ and standard deviation σ of F's accuracy in 40 experiments[11] for each scenario. The accuracies obtained by F for all these scenarios are present in Table 5.

Although the presence of MagNet has provided some sort of protection to F (when compared to the results belonging to Scenario (i)), it becomes notorious, by comparing the results in Table 5 related to Scenarios (ii) and (iii), that MultiMagNet has also overcome MagNet when the performance of F is taken into account, thus providing to F a better protection than MagNet. It is also worth mentioning that the Reformer step has had significant importance on improving even more the performance of F, as it can be seen by the results related to Scenario (v). This may be related to the fact that, even with the adoption of an ensemble for detecting perturbations, few adversarial images still might have been able to bypass the MultiMagNet's detection step and fool F. So, the Reformed step has behaved as an additional protection to the application classifier. Therefore, the results in Tables 2, 3 and 5 emphasise that MultiMagNet has been more effective than MagNet in detecting adver-

[10]The word ”Reformer” refers to the Reform Image process, as illustrated by Figure 4 and explained in Section 4.2.5.

[11]It has been formed a new ensemble R for every 100 input images belonging to D set, in a way similar to the experiments which produce the results in Tables 2 and 3.

sarial images and protecting the application classifier, thus providing further significant evidences that the hypothesis raised by this work is true.

6 FINAL CONSIDERATIONS

In recent years, several work have demonstrated that Deep Neural Networks are susceptible to be intentionally induced to misclassification by adversarial images (*i.e.* images which contain perturbations, usually imperceptible to human eyes), fact that which precludes the application of these learning algorithms in several security-critical decision support systems (Klarreich, 2016). Although various defenses have been proposed against adversarial images, most of them have already been bypassed by allowing the attacker to easily map their inner behaviour. As a means of reducing the behaviour predictability, a research has aimed to create a non-deterministic detection method called MagNet (Meng and Chen, 2017). However, recent studies reveal MagNet can also be bypassed by adversarial images (Carlini and Wagner, 2017b).

Before this alarming scenario, the present work has raised the hypothesis the insertion of multiple defense components, randomly selected, in the detection method can amplify the non-deterministic effect and thus making the defense more robust than MagNet before different attack algorithms. Therefore, this work has introduced MultiMagNet, a method for detecting adversarial images which makes use of multiple defense components (implemented as autoencoders), selected at random and arranged in ensembles for decision making. The experimental results performed on images from MNIST and CIFAR-10 dataset point to the veracity of the raised hypothesis, showing that MultiMagNet has overcome MagNet (the MultiMagNet's version which randomly chooses only one autoencoder) in most of the evaluated scenarios. In summary, the following main contributions of this work are:

- The development of MultiMagNet, a non-deterministic defense based of ensembles for detecting adversarial images in decision support systems;
- The accomplishment of the first[12] comparative study with MagNet, in order to validate the hypothesis raised by this work;
- The online availability of all the implementation and needed resources for reproducing the experiments.

[12]Regarding the best of the authors' knowledge.

For future work, it can be highlighted: (i) the research of novel non-deterministic architectures for detecting and classifying adversarial images, (ii) the implementation of novel techiniques to automatically define classification thresholds, (iii) the adoption of different techniques for preprocessing images.

ACKNOWLEDGMENTS

The present work has been conducted with the support of the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) - Finacing Code 001.

REFERENCES

Bae, G., Lee, H., Son, S., Hwang, D., and Kim, J. (2018). Secure and robust user authentication using partial fingerprint matching. In *Consumer Electronics (ICCE), 2018 IEEE International Conference on*, pages 1–6. IEEE.

Carlini, N. and Wagner, D. (2017a). Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. In *10th ACM Workshop on Artificial Intelligence and Security*, page 12, Dallas, TX.

Carlini, N. and Wagner, D. (2017b). Magnet and" efficient defenses against adversarial attacks" are not robust to adversarial examples. *arXiv preprint arXiv:1711.08478*.

Carlini, N. and Wagner, D. (2017c). Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE.

Derawi, M. O., Nickel, C., Bours, P., and Busch, C. (2010). Unobtrusive user-authentication on mobile phones using biometric gait recognition. In *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2010 Sixth International Conference on*, pages 306–311. IEEE.

Ding, L., Fang, W., Luo, H., Love, P. E., Zhong, B., and Ouyang, X. (2018). A deep hybrid learning model to detect unsafe behavior: integrating convolution neural networks and long short-term memory. *Automation in Construction*, 86:118–124.

Gong, Z., Wang, W., and Ku, W.-S. (2017). Adversarial and clean data are not twins. *arXiv preprint arXiv:1704.04960*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Goodfellow, I., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *2016 IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778. IEEE.

He, W., Wei, J., Chen, X., Carlini, N., and Song, D. (2017). Adversarial Example Defenses: Ensembles of Weak Defenses are not Strong. In *11th USENIX Workshop on Offensive Technologies (WOOT' 17)*, Vancouver, CA.

Hendrycks, D. and Gimpel, K. (2017). Early methods for detecting adversarial images. *Workshop track -ICLR 2017*.

Hu, J., Shen, L., and Sun, G. (2017). Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*.

Kannan, H., Kurakin, A., and Goodfellow, I. (2018). Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*.

Karpathy, A. (2014). What I learned from competing against a ConvNet on Imagenet. Available at http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet. Accessed in September 02, 2018.

Klarreich, E. (2016). Learning securely. *Communications of the ACM*, 59(11):12–14.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images.

Kurakin, A., Goodfellow, I., and Bengio, S. (2016a). Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*.

Kurakin, A., Goodfellow, I., and Bengio, S. (2016b). Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*.

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

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Meng, D. and Chen, H. (2017). Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 135–147. ACM.

Metzen, J. H., Genewein, T., Fischer, V., and Bischoff, B. (2017). On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*.

Moosavi-Dezfooli, S.-M., Fawzi, A., and 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*, pages 2574–2582.

Nicolae, M.-I., Sinn, M., Tran, M. N., Rawat, A., Wistuba, M., Zantedeschi, V., Baracaldo, N., Chen, B., Ludwig, H., Molloy, I., and Edwards, B. (2018). Adversarial robustness toolbox v0.3.0. *CoRR*, 1807.01069.

Obermeyer, Z. and Emanuel, E. J. (2016). Predicting the future — big data, machine learning, and clinical medicine. *The New England journal of medicine*, 375(13):1216.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. (2017). Practical Black-Box Attacks against Machine Learning. In *ACM Asia Conference on Computer and Communications Security (ASIACCS)*, pages 506–519.

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

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. (2016b). Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. In *Proceedings - 2016 IEEE Symposium on Security and Privacy, SP 2016*, pages 582–597.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*.

Srivastava, S., Priyadarshini, J., Gopal, S., Gupta, S., and Dayal, H. S. (2019). Optical character recognition on bank cheques using 2d convolution neural network. In *Applications of Artificial Intelligence Techniques in Engineering*, pages 589–596. Springer.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. In *International Conference on Learning Representations*, pages 1–10.

Tolosana, R., Vera-Rodriguez, R., Fierrez, J., and Ortega-Garcia, J. (2018). Exploring recurrent neural networks for on-line handwritten signature biometrics. *IEEE Access*, 6(5128-5138):1–7.

Xu, W., Evans, D., and Qi, Y. (2018). Feature squeezing: Detecting adversarial examples in deep neural networks. *Network and Distributed Systems Security Symposium (NDSS) 2018*.

Zantedeschi, V., Nicolae, M.-I., and Rawat, A. (2017). Efficient defenses against adversarial attacks. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 39–49. ACM.