FEDERAL UNIVERSITY OF SANTA CATARINA
TECHNOLOGICAL CENTER
GRADUATION PROGRAM IN AUTOMATION AND SYSTEMS ENGINEERING

João Gabriel Zago

Defense Methods for Convolutional Neural Networks Against Adversarial Attacks

Florianópolis
2021

João Gabriel Zago

Defense Methods for Convolutional Neural Networks Against Adversarial Attacks

Thesis submitted to the Graduation Program in Automation and Systems Engineering from the Federal University of Santa Catarina for obtaining the Master Degree in Automation and Systems Engineering.
Supervisor:: Prof. Fabio Luis Baldissera, Dr.
Co-supervisor:: Prof. Rodrigo Tacla Saad, Dr., Prof. Eric Aislan Antonelo, Dr.

Florianópolis
2021

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Zago, João Gabriel
Defense Methods for Convolutional Neural Networks Against Adversarial Attacks / João Gabriel Zago ; orientador, Fabio Luis Baldissera, coorientador, Rodrigo Tacla Saad, coorientador, Eric Aislan Antonelo, 2021.
66 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Engenharia de Automação e Sistemas, Florianópolis, 2021.

Inclui referências.

1. Engenharia de Automação e Sistemas. 2. Adversarial examples. 3. Convolutional Neural Networks. 4. Adversarial defenses. I. Baldissera, Fabio Luis. II. Tacla Saad, Rodrigo. III. Aislan Antonelo, Eric IV. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. V. Título.

João Gabriel Zago

Defense Methods for Convolutional Neural Networks Against Adversarial Attacks

O presente trabalho em nível de mestrado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof. Maurício Fernandes Figueiredo, Dr.
Federal University of São Carlos

Prof. Jomi Fred Hubner, Dr.
Federal University of Santa Catarina

Prof. Marcelo Stemmer, Dr.
Federal University of Santa Catarina

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de mestre em Engenharia de Automação e Sistemas.

Documento assinado digitalmente
Werner Kraus Junior
Data: 04/05/2021 21:27:46-0300
CPF: 531.085.239-53
Verifique as assinaturas em https://v.ufsc.br

Prof. Werner Kraus Junior, Dr.
Coordenador do Programa

Documento assinado digitalmente
Fabio Luis Baldissera
Data: 05/05/2021 10:06:31-0300
CPF: 036.009.569-06
Verifique as assinaturas em https://v.ufsc.br

Prof. Fabio Luis Baldissera, Dr.
Supervisor:

Florianópolis, 5 de Abril de 2021.

I dedicate this Thesis to my parents, my sisters, Dori and all the loved ones. Hope you appreciate this work.

ACKNOWLEDGEMENTS

Working on this Master's Thesis and, in general, developing research in this scientific environment is arduous. Without the support of various people, the difficulty level of this path would have increased substantially. First of all, I wish to express my sincere appreciation to my Supervisor, Prof. Fabio Baldissera, Dr. and Co-supervisors Prof. Rodrigo Tacla Saad, Dr. and Prof.Eric Aislan Antonelo, Dr. for their support, intellectual guidance, encouragement, and valuable advice. This work would have significantly different results, and this document would not have reached this level of coherence without their assistance and teachings.

My appreciation also goes for the Coordination for the Improvement of Higher Education Personnel (CAPES) for offering a scholarship as a graduate student, enabling access to higher education. Last but not least, I would like to express my appreciation for my parents, Mr. Alair Aparecido Zago and Madam Maria Lucia Celestino Zago, for their emotional and financial support, providing all the necessary tools and the opportunity of going through this experience.

RESUMO

Mesmo com seu sucesso na classificação de imagens, as redes neurais convolucionais são frágeis com relação a pequenas perturbações inseridas nas imagens que tal modelo deve classificar: pequenas alterações nos valores de alguns dos *pixels* da sua entrada podem resultar em uma classificação de saída completamente diferente. Tais imagens intencionalmente perturbadas para enganar o classificador são conhecidas como exemplos adversários. A vulnerabilidade das redes neurais convolucionais com respeito aos exemplos adversários levanta um alerta com relação a utilização destes modelos em aplicações que necessitam de garantias de segurança: que envolvem risco a vida, ambiental, ou tem implicações financeiras. Esta dissertação contém dois métodos complementares e computacionalmente baratos que buscam auxiliar a aliviar e eliminar tal vulnerabilidade. a) uma nova estratégias que reduz a efetividade de ataques adversários, ofuscando as saídas da rede neural a partir da adição de perturbações controladas, não necessitando de nenhum tipo de treinamento; e b) um método que emprega a lei de Benford para distinguir imagens sem perturbação de exemplos adversários, provendo uma proteção extra que age nas entradas de um classificador vulnerável. O primeiro método de defesa desenvolvido (a) não somente reduz a taxa de sucesso, mas também força a adição de uma perturbação de maior magnitude por parte do atacante. O estudo conduzido em (b) indicou que: 1) imagens adversárias possuem uma tendência de desviar de forma significativa com respeito a lei de Benford, em comparação com imagens que não foram perturbadas; 2) há um incremento deste desvio com o aumento da perturbação inserida; 3) em alguns casos é possível identificar ataques em andamento através de um monitoramento deste desvio, o que torna possível o desligamento do atacante antes que o mesmo complete a sua operação e crie um exemplo adversário. Por fim, pelo fato de ambos os métodos propostos serem ortogonais, é esperada uma maior proteção contra ataques adversários ao se utilizar ambos simultaneamente.

Palavras-chave: Redes neurais convolucionais. Métodos de defesa. Ataques adversários. Lei de Benford. Detecção de exemplos adversários.

RESUMO EXPANDIDO

Introdução

As redes neurais convolucionais alcançaram o estado da arte na classificação de imagens (KRIZHEVSKY; SUTSKEVER; HINTON, 2012; SZEGEDY; VANHOUCKE, *et al.*, 2016; SIMONYAN; ZISSERMAN, 2014). Entretanto, estes modelos se mantém vulneráveis com relação a pequenas perturbações inseridas nas entradas que estas classificam, o que restringe seu uso em aplicações que não necessitam de garantias de segurança (SZEGEDY; ZAREMBA, *et al.*, 2013; HUANG *et al.*, 2017). Para um melhor entendimento deste fenômeno, pesquisadores da área desenvolveram algoritmos para a geração dos chamados exemplos adversários, que representam tais entradas projetadas para induzir um erro de classificação por parte do modelo de classificação, sem alterar a classe atribuída por um ser humano.

Foram propostas e desenvolvidas diversas abordagens para proteger as redes convolucionais contra tais algoritmos, as quais podem ser classificadas em dois grandes grupos: a) centradas na redes, as quais buscam reduzir a vulnerabilidade da redes com relação aos exemplos adversários atuando no modelo de classificação (MADRY *et al.*, 2017; PAPERNOT; MCDANIEL; WU, *et al.*, 2016; KATZ *et al.*, 2017); b) centradas nas entradas, que são aquelas que atuam nas entradas dadas ao modelo, detectando ou reconstruindo-as de tal forma a reduzir o impacto destes ataques (LU; ISSARANON; FORSYTH, 2017; SONG *et al.*, 2017). Mesmo com os avançados das técnicas de defesa contra ataques adversários, até o momento os atacantes seguem na frente, o que parece ser resultado de uma vulnerabilidade inerente dos modelos de classificação baseados em redes neurais convolucionais, com respeito a imagens perturbadas que se encontram fora da distribuição de exemplos de treinamento do modelo.

Objetivos

Com base nesta fragilidade apresentada por modelos de classificação de imagens, o objetivo principal desta dissertação foi trabalhar no desenvolvimento de métodos de defesa visando uma redução no impacto dos ataques adversários, fazendo com o que o número de exemplos adversários gerados fosse reduzido. Como objetivos específicos deste trabalho, buscou-se realizar um estudo detalhado da literatura de ataques e defesas para redes neurais com respeito aos exemplos adversários, propor uma nova abordagem para defender as redes neurais convolucionais, avaliar o método proposto frente ao que já existe na literatura, realizar alguns estudos teóricos em exemplos simplificados para criar um entendimento maior sobre o problema em questão, propiciando assim um aprimoramento e uma busca por novas técnicas de defesa.

Metodologia

Para o desenvolvimento da pesquisa, iniciou-se com uma investigação detalhada da literatura de exemplos adversários relacionados a redes neurais convolucionais, já que tais modelos representam atualmente o estado da arte em classificação de imagens. Dentro desta busca, foi realizada a implementação de métodos de ataque e defesa existentes na literatura com o intuito de identificar possíveis melhorias e oportunidades para a proposição e o desenvolvimento de novas abordagens. Com base nas informações obtidas, a etapa seguinte consistia de uma primeiro proposta de método de

defesa para os modelos de classificação. Para avaliar o funcionamento da ideia, foi construída uma estrutura de teste composta por um modelo convolucional treinado com os dados do MNIST, que são imagens de baixa resolução contendo dígitos de 0 até 9, além de ataques adversário. Validado o método para este modelo, seria então iniciada uma nova etapa de exploração para identificar melhorias na proposta e possíveis novas abordagens para a redução do impacto de exemplos adversários no funcionamento das redes neurais.

Resultados e Discussões

A duas contribuições principais construídas com a realização desta pesquisa foram: 1) um método de defesa para redes neurais convolucionais contra métodos de ataque do tipo caixa-preta (aqueles que necessitam de acesso apenas às saídas da rede), que se baseia na aplicação de um ruído aleatório na distribuição de saída do modelo; e 2) uma análise estatística da distribuição do primeiro dígito que permite a detecção de exemplos adversários após serem gerados ou durante o seu procedimento de construção.

O método (1) consiste da adição de uma perturbação na distribuição de saída dada pela rede neural, cujo o objetivo é o de confundir o algoritmo de ataque com relação a direção da perturbação a ser inserida. Tal proposta não envolve nenhum processo de treinamento da rede neural, atuando apenas durante o período de inferência do modelo. O método (1): a) reduz a taxa de sucesso de um ataque caixa-preta particular de 99.8% para 65.3%; e b) induz um aumento na magnitude de perturbação inserida pelo atacantes, o que é um resultado importante, já que um bom ataque adversário é aquele que gera exemplos adversários com perturbações aproximadamente imperceptíveis.

Relativa a proposta (2), a análise apresentada permite o desenvolvimento de um novo método de detecção de exemplos adversários com base na lei de Benford (BENFORD, 1938). Tal lei apresenta o comportamento esperado para a distribuição do primeiro dígito de um conjunto de dados não artificiais. Analisando a distribuição do primeiro dígito em imagens em comparação com a distribuição teórica prevista na lei de Benford, é possível identificar imagens propositalmente alteradas. Os resultados dos experimentos realizados mostraram que: a) imagens adversários, diferentemente de imagens naturais, tendem a desviar significativamente da lei de Benford; b) o desvio é maior para algoritmos de ataque que se baseia na utilização da $||.||_\infty$-norma; c) o aumento da magnitude da perturbação faz com que o desvio com relação a lei de Berford aumente; d) em alguns casos, é possível identificar atacantes que estejam no processo de geração de exemplos adversários, antes mesmo de que o ataque seja finalizado; e e) tal abordagem provê uma métrica de baixo custo computacional que pode ser utilizada para a detecção de exemplos adversários.

Considerações Finais

Neste trabalho foram propostas duas abordagens diferentes para lidar com os exemplos adversário, ambas com foco em reduzir a vulnerabilidade de tais modelos frente a diferentes ataques adversários. O primeiro método consiste da adição de uma perturbação aleatório e controlada nas camada de saída de um classificador neural. Tal abordagem consiste de uma estratégia de defesa com foco em ataque do tipo

caixa-preta, dificultando a identificação dos *pixels* que levarão a uma mudança de classificação. A segunda proposta consiste de uma análise que pode ser utilizada para a criação de um detector de exemplos adversários. Esta abordagem se baseia no calculo da distribuição do primeiro dígito para as entradas do classificador, tomando como premissa a ideia de que exemplos adversários não seguem a distribuição prevista na lei de Benford. Ambas propostas trabalham apenas na fase de inferência, não necessitando de qualquer tipo de alteração ou incremente no processo de treinamento ou alteração nos parâmetros e na arquitetura da rede.

ABSTRACT

Despite its success in image classification, Convolutional Neural Networks (CNN) are still fragile to small perturbations in the input images they have to classify: slight changes in the values of some pixels might result in completely different network outputs. Such images purposefully perturbed to deceive a classifier are known as adversarial images. This vulnerability of CNN to adversarial images raises concerns in safety-sensitive applications: involving life-threatening, environmental, or financial implications. This thesis proposes two computationally cheap and complementary methods to help circumvent and alleviate this fragility of CNN: a) a novel strategy that reduces the success of adversarial attacks by obfuscating the softmax output, which does not require any network training; and b) a method that employs Benford's Law for distinguishing transformed natural images from transformed adversarial ones at the pixel level, providing an extra shield acting at the input layer of vulnerable CNN. The defense we developed in (a) not only decreases the attack success rate but also forces the attack algorithm to insert larger perturbations in the input images. The study conducted in (b) indicates that: 1) adversarial images tend to deviate significantly more from Benford's distribution than unaltered images; 2) this deviation increases with the magnitude of the perturbation; 3) in some cases, it is possible to identify ongoing attacks by online monitoring this deviation, making it possible to turn off the classifier for the particular requester before it completes an attack. Finally, these two methods are orthogonal in that we expect the CNN classifier to get better protection against attacks while using them simultaneously.

Keywords: Convolutional neural networks. Defense strategies. Adversarial attacks. Benford's law. Adversarial detection.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Convolutional Neural Networks (CNN) have achieved state-of-the-art performance in image classification (KRIZHEVSKY; SUTSKEVER; HINTON, 2012; SZEGEDY; VANHOUCKE, *et al.*, 2016; SIMONYAN; ZISSERMAN, 2014). However, they are vulnerable to small perturbations in the inputs they ought to classify, a fact that restricts them to applications that are not safety-sensitive (SZEGEDY; ZAREMBA, *et al.*, 2013; HUANG *et al.*, 2017). One example of such fragility is that of a neural network that mistakes a slightly perturbed image of a panda for a gibbon (GOODFELLOW; SHLENS; SZEGEDY, 2014). To better study this vulnerability phenomenon, researchers have recently developed algorithms that generate the so-called adversarial examples. These are inputs purposefully designed to induce a misclassification by the neural network (though not by a human being).

One can divide the adversarial attack algorithms into two main categories regarding the attacker's knowledge about the network: white-box and black-box methods. White-box attacks are those that use at least one internal parameter of the neural network (e.g., its number of layers or the values of its synaptic weights) to generate their attacks (SZEGEDY; ZAREMBA, *et al.*, 2013; GOODFELLOW; SHLENS; SZEGEDY, 2014; KURAKIN; GOODFELLOW; BENGIO, 2016a; PAPERNOT; MCDANIEL; JHA, *et al.*, 2015; MOOSAVI-DEZFOOLI; FAWZI; FROSSARD, 2016; CARLINI; WAGNER, 2017b; MOOSAVI-DEZFOOLI; FAWZI; FAWZI, *et al.*, 2017; SABOUR *et al.*, 2015; MADRY *et al.*, 2017). On the other hand, black-box algorithms only access the outputs of CNN classifiers for a given set of input images to devise adversarial examples (PAPERNOT; MCDANIEL; GOODFELLOW, *et al.*, 2017; NARODYTSKA; KASIVISWANATHAN, 2016; CHEN *et al.*, 2017; SU; VARGAS; SAKURAI, 2017; ZHAO; DUA; SINGH, 2018).

To protect CNN against such attacks, researchers have devised a wide range of defense strategies in the last few years. We can group them into two major categories: a) network-centered, whose aim is to decrease the neural network vulnerability to adversarial attacks (SZEGEDY; ZAREMBA, *et al.*, 2013; GOODFELLOW; SHLENS; SZEGEDY, 2014; MADRY *et al.*, 2017; TRAMÈR *et al.*, 2018; KURAKIN; GOODFELLOW; BENGIO, 2016b; PAPERNOT; MCDANIEL; WU, *et al.*, 2016; MOOSAVI-DEZFOOLI; FAWZI; FROSSARD, 2016); b) input-centered, where the goal is to detect or reconstruct adversarial images (LU; ISSARANON; FORSYTH, 2017; METZEN *et al.*, 2017; FEINMAN *et al.*, 2017; GROSSE *et al.*, 2017; SONG *et al.*, 2017; KATZ *et al.*, 2017; GU; RIGAZIO, 2014; SONG *et al.*, 2017; MENG; CHEN, 2017). Despite the advances in defense techniques, the attackers are, up to now, winning this digital arms race since CNN-based classifiers seem to be inherently vulnerable to perturbed images that lie outside their training set probability distributions.

In this Master thesis, we propose two different approaches that act on the defensive side of neural network classifiers. The first one reduces the attack success rate by inserting controlled disturbances to the classifiers' output probabilities (network-centered). The second provides a different feature for detecting adversarial attacks requiring only the given input without changing the CNN (input-centered).

1.1 CONTRIBUTIONS

Our research brings two main contributions:

1. a disturbance based defense method against black-box attacks that need access to the output layer of CNN (class probabilities);
2. a statistical analysis that enables detecting adversarial examples prior to or during an ongoing attack.

The method referred in (1) consists of disturbing the class probabilities output by CNN to confuse the attack algorithm regarding the promising perturbation directions (i.e., pixels that, when perturbed, render an adversarial example). Note that this strategy is different from adversarial training (SZEGEDY; ZAREMBA, *et al.*, 2013; GOODFELLOW; SHLENS; SZEGEDY, 2014; MOOSAVI-DEZFOOLI; FAWZI; FROSSARD, 2016; MADRY *et al.*, 2017; KURAKIN; GOODFELLOW; BENGIO, 2016b; TRAMÈR *et al.*, 2018) and defensive distillation (PAPERNOT; MCDANIEL; WU, *et al.*, 2016), which requires training CNN regarding specific criteria, resulting in an increment of the classifier's robustness (i.e., reducing the effective number of existing adversarial images). Our approach does not involve training CNN, but it acts at inference time, disturbing in a controlled way the predicted class probabilities, rendering it a computationally cheap method compared to the literature. What our strategy does is to make it harder for the attack algorithm to find such adversarial inputs. As we show later, our proposal: a) reduces the success rate of particular black-box attacks from 99.8% to 65.3%; and b) induces the attack algorithms to insert perturbations of higher magnitude. This result is an interesting feature because a good adversarial image is one that not only deceives the classifier but does so "gracefully".

Our method stands out in comparison to adversarial training and defensive distillation because: one showed that scaling up the size of the training set using adversarial training method, seems to be unfeasible as the authors proved to be insufficient to improve the neural network robustness against all possible adversarial examples (KHOURY; HADFIELD-MENELL, 2019). Yet, the defensive distillation succumbed to the C&W adversarial attack (CARLINI; WAGNER, 2017b), which differs from ZOO by the gradient calculation procedure (CHEN *et al.*, 2017).

Regarding the second contribution, we present a thorough analysis that could enable the development of a detection method for adversarial examples based on Benford's law (BENFORD, 1938). This law states the behavior of the first digit distribution in numbers from real-world data sources. By analyzing the first digit distribution of the pixels in images concerning Benford's law theoretical distribution, it is possible to infer purposefully altered images. The application of Benford's Law (BL) came from its uses in other domains, such as fraud detection (TÖDTER, 2009; DECKERT; MYAGKOV; ORDESHOOK, 2011) and image forensics (MILANI *et al.*, 2016; PEVNY; FRIDRICH, 2008). Our empirical experiments show that: 1) adversarial images, differently from natural ones, tend to deviate significantly from BL; 2) this deviation is higher for attack algorithms based on $||.||_\infty$-norm perturbations; 3) deviations from Benford's Law increase with the magnitude of the designed perturbation; 4) in some cases, adversarial attacks can be anticipated even before the perturbed image becomes adversarial, that is, a deviation from BL takes place during the perturbation iterative design process; and, 5) another fundamental characteristic of this new approach is that it produces a computationally cheap low-dimensional input feature that could be used for adversarial image detection.

Besides, both of the proposed procedures are orthogonal among existing defense methods as their functioning does not influence the training procedure or the architecture of neural networks classifier. This feature enables the combination of these approaches with different adversarial defense methods.

1.2 DOCUMENT ORGANIZATION

In this section, we present the document structure and guidance for reading paths regarding the lector profile. Figure 1 presents three different recommended reading tracks that one may follow for reading this document, composed of five chapters. Chapter 1 started with a presentation on the context and the contributions of this research.

In Chapter 2 we present fundamental concepts on adversarial examples regarding artificial neural networks for image classification.

Chapter 3 shows the first method proposed in this research based on the application of disturbance to the output probabilities of a neural network.

In Chapter 4 we show our second proposed approach, comprising a statistical analysis of the first digit distribution of images regarding Benford's Law. As chapters 3 and 4 present independent (although complementary to each other) methods, their reading order choice is up to the reader.

Finally, Chapter 5 shows the conclusions from our methods regarding the achieved results. Furthermore, we describe future works and references to the papers submitted during this Master's research.

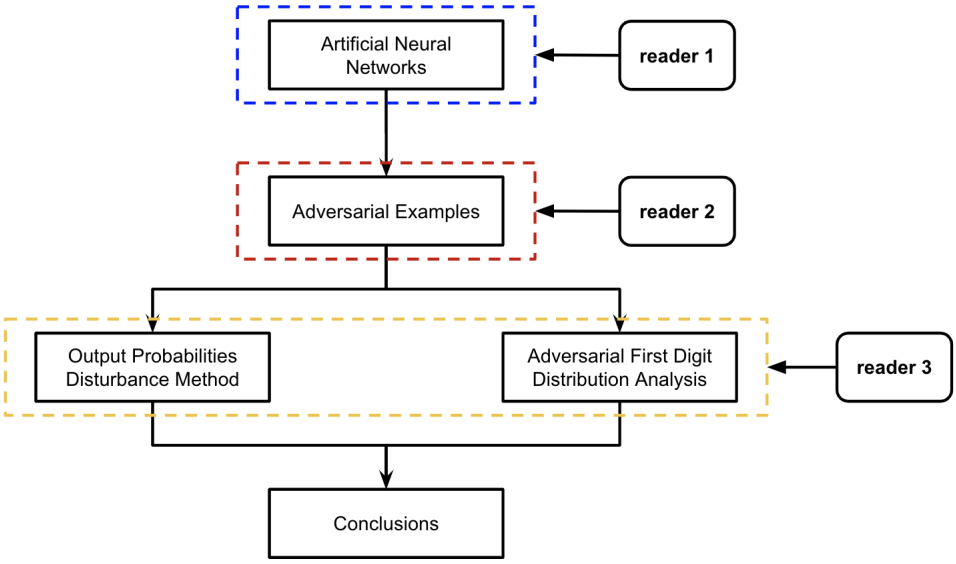


Figure 1 – Flowchart presenting three different reading paths, regarding different interests. The reader 1 has inexperienced comprehension on artificial neural networks. Reader 2 has the necessary knowledge on artificial and convolutional neural networks, yet not a deeper understanding on adversarial examples. The last reader, number 3, comprehends the adversarial examples theory and focus the proposed approaches.

2 ADVERSARIAL EXAMPLES: FUNDAMENTALS

In this chapter, we begin introducing artificial neural networks. We expose information about their inspiration, different architectures (i.e., fully connected and convolutional neural networks), and their inference and training processes. Further, we describe and define the adversarial examples. We present some of their properties and the concern raised by applying these models in safety-sensitive environments. We also show some of the attack procedures proposed in the literature for generating those examples and also defending neural networks against them.

2.1 ARTIFICIAL NEURAL NETWORKS

Architecture

Artificial neural networks are mathematical models for information processing. They draw their inspiration from the architecture and functioning of animal brains. As their (much more intricate) biological counterparts, artificial neural networks find applications in different domains, such as image classification (KRIZHEVSKY; SUTSKEVER; HINTON, 2012; SIMONYAN; ZISSERMAN, 2014; SZEGEDY; VANHOUCKE, *et al.*, 2016), natural language processing (HINTON; DENG, *et al.*, 2012) and many others.

An artificial neuron is the processing unity of information for the operations of a neural network. Figure 2 presents a visual representation of a generic artificial neuron functioning. A single neuron comprises three elementary components: the synaptic

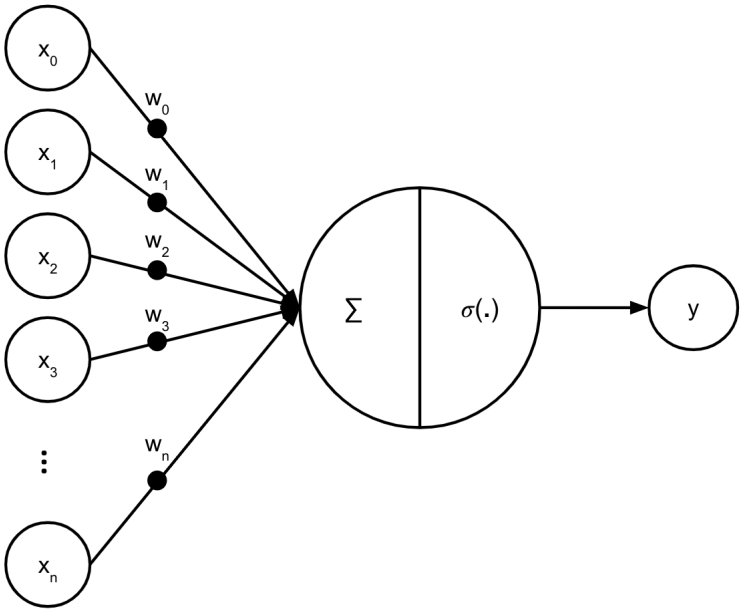


Figure 2 – Artificial neuron functioning representation. The inputs are weighted, summed and activated by a function σ. The output comprises of the activation from a linear combination of the inputs.

weights applied to each input, the sum operation of the weighted inputs, and the activation function. Equation 1 shows the mathematical representation for an artificial neuron mapping.

$$y = \sigma \left(\sum_{i=0}^{n} w_i x_i \right) \tag{1}$$

Artificial networks are composed of several layers, which, in turn, have several neurons. There are different architecture arrangements for the neurons inside an artificial neural network. We present here the fully connected and the convolution architectures. The latter achieves state-of-the-art results for image classification (KRIZHEVSKY; SUTSKEVER; HINTON, 2012; SIMONYAN; ZISSERMAN, 2014; SZEGEDY; VANHOUCKE, *et al.*, 2016), and also is the scope of this research.

The fully connected architecture comprises arranging the neurons so that all the artificial neural cells from the previous layer connect directly with all the neurons from the posterior layer. The artificial neuron: a) receives inputs from some (possibly all) neighboring neurons located in the layer $k-1$; b) map these inputs to a scalar output, according to a (generally nonlinear) function, called *activation function*; and c) sends its output signal to some (possibly all) neurons in the layer $k+1$. First layers neurons receive external inputs, which can be images, sounds, or other data objects. Figure 2 presents this arrangement. Neural networks with more than two layers (i.e., the input and output layers) include the so-called hidden layers, which are those internal layers connecting a neural network's inputs and outputs.

There are two main processes assigned to artificial neural networks: inference and training. Formally, one can model artificial neural networks inference process by mathematical maps $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ that go from a given n-dimensional input space to some m-dimensional output space. In the case of image classification, for instance, $x \in \mathbb{R}^n$ is the mathematical representation of an image, whereas $F(x) \in \mathbb{R}^m$ is a probability vector, where $F(x)_i$ contains the probability that the example x belongs to

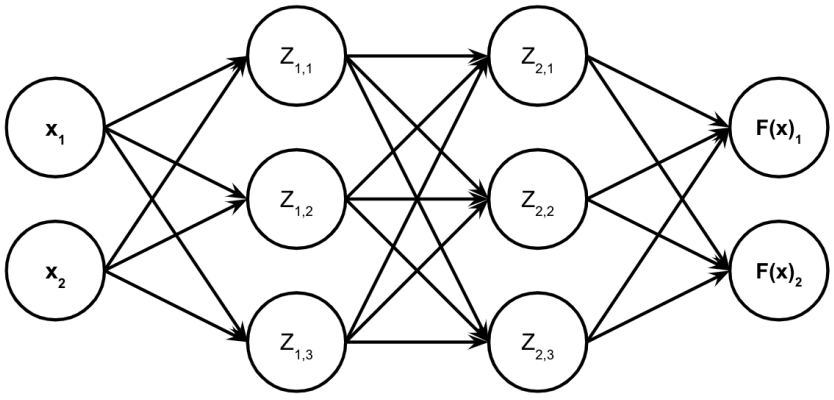


Figure 3 – Directed acyclic graph representation of a fully connected neural network. Each internal node represents a neuron, represented in Figure 2

the class i, $1 \leq i \leq m$, e.g., a cat, a dog or a baseball bat.

Let Z_k denote the output of the k-th layer of a neural network N, where $Z_0 = x$ is the external input and $Z_{L+1} = F(x)$ is the network's output, with L being the number of hidden layers. Then, for the case of a fully connected neural network, Equation 2 describes the map between layers, where $W_k \in \mathbb{R}^{|Z_{k-1}| \times |Z_k|}$ is the weights matrix, $b_k \in \mathbb{R}^{|Z_k|}$ the bias vector, and $f_k : \mathbb{R}^{|Z_k|} \rightarrow \mathbb{R}^{|Z_k|}$ the activation function of a single layer. The weights and biases are the adjustable parameters.

$$Z_k = f_k(W_k \times Z_{k-1} + b_k) \tag{2}$$

The choice of the activation functions, as well as the network architecture, are application-dependent. For instance, classification models use the softmax (or sigmoid for binary classifier) function as activation for the output layer, as this function returns the probability distribution associated with each class.

Training

The training process in supervised learning comprises adapting the weights of an artificial neural network based on a training data set. This data set consists of input patterns and their respective target outputs. We define the training set as $\{(x^{(j)}, t^{(j)}) : x^{(j)} \in \mathbb{X} \text{ and } t^{(j)} \in \mathbb{T}, \forall j \in \{1, \ldots, T\}\}$, where $\mathbb{X} \subseteq \mathbb{R}^n$ is a subset of the input space (i.e., the set of training inputs), $\mathbb{T} \subseteq \mathbb{R}^m$ the set of expected outputs or targets, $T = |\mathbb{X}| = |\mathbb{T}|$ represents the size of the training set and $t^{(j)}$ the corresponding expected output associated with $x^{(j)}$. The epochs of training indicate the number of iterations over all the samples from the data set. The training procedure consists of an optimization task to minimize a given loss function, regarding the adjustable parameters $\mathbf{W}$ and $\mathbf{b}$, that represent W_k and b_k for all $k \in \{1, \ldots, L\}$. Without loss of generality, we represent both, $\mathbf{W}$ and $\mathbf{b}$, as $\mathbf{W}'$.

The loss function, denoted as J, must agree with the problem solved by the neural network. For classification models, the categorical cross-entropy is preferable, presented in Equation 3:

$$J_{CE}(\mathbf{W}') = -\sum_{j}^{T}\sum_{i}^{m} t_i^{(j)} \log(F(x^{(j)}; \mathbf{W}')_i) \tag{3}$$

where T denotes the size of the training set and $t^{(j)}$ the desired output for $x^{(j)}$. On the other hand, the mean squared error usually fits better for regression models, showed in Equation 4:

$$J_{MSE}(\mathbf{W}') = \frac{1}{T}\sum_{j}^{T}\sum_{i}^{m}(t_i^{(j)} - F(x^{(j)}; \mathbf{W}')_i)^2 \tag{4}$$

The gradient descent and the Adam optimizer (KINGMA; BA, 2014) are examples of optimization algorithms usually employed to minimize the loss function. Equation 5

shows the weights update by using the gradient descent optimization procedure:

$$\mathbf{W}' = \mathbf{W}' - \eta \times \frac{\partial J(\mathbf{W}')}{\partial \mathbf{W}'} \tag{5}$$

where $\mathbf{W}'$ represent the weights and bias matrices, and η represents the learning rate of the training process.

The minimization relies on the gradient of the loss function regarding the weights, $\nabla J(\mathbf{W}')$, which employs the whole data set or a minibatch of examples. The latter form is called stochastic gradient descent. The computation of gradients takes place through the error back-propagation algorithm (RUMELHART; HINTON; WILLIAMS, 1986).

2.2 CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Networks (CNN) are Artificial Neural Networks specialized in processing data with a grid-like topology. This architecture comprises two distinct operations: the discrete convolution and pooling operations. Typically, CNN comprise two types of layers, where each of them represents a different operation (i.e., convolutional and pooling layers). Discrete convolutional layers return the activation mapping of a discrete convolution output, also known as feature maps. The pooling layer locally summarizes the output of the activation function.

The discrete convolution operation extracts spatial information from a given input and propagates it to the following layers, as presented in Equation 6:

$$Z_k(i,j) = f_k\left(\left(\sum_{o}^{O}\sum_{p}^{P} Z_{k-1}(i-o, j-p) \times K_k(o,p)\right) + b_k(i,j)\right) \tag{6}$$

where K is the kernel from the convolution operation, i and j are matrix indices, O and P represent the size of the kernel employed in the convolution operation.

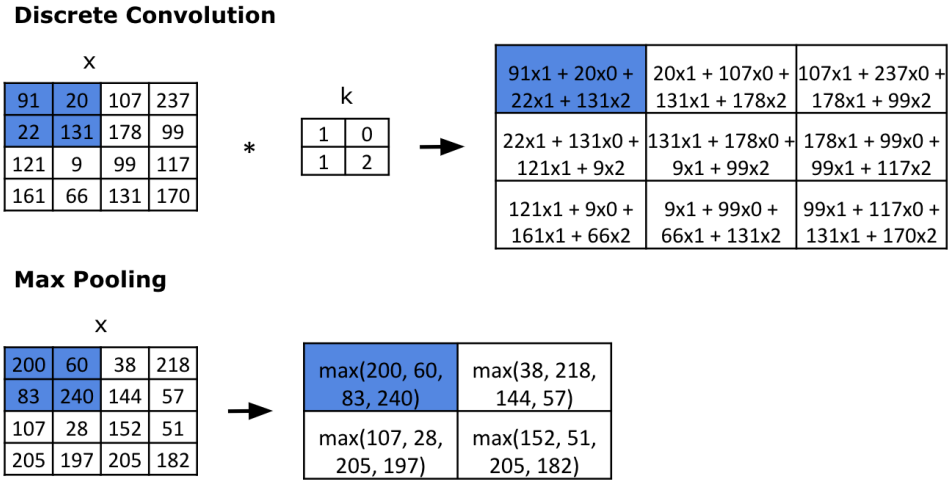


Figure 4 – Functioning examples of both, the discrete convolution and the max pooling operation.

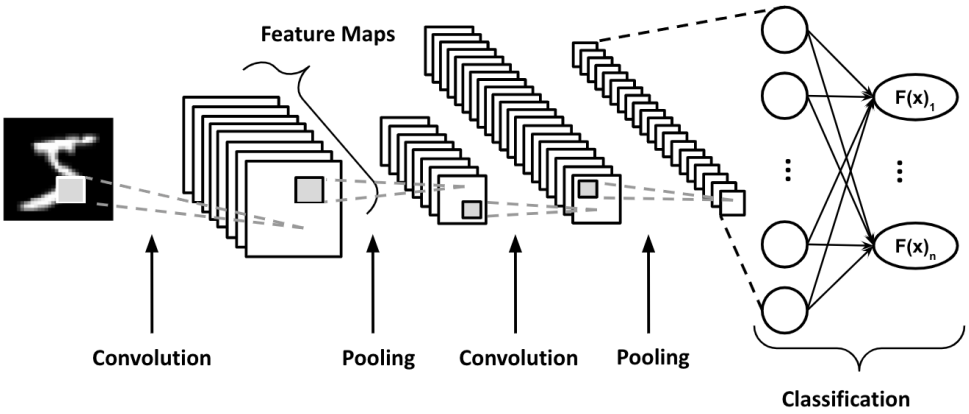


Figure 5 – Representation of a Convolutional Neural Network. The first and second layers correspond to convolution and a pooling operation, respectively. The following layers represent a Fully Connected Neural Network classifier.

The pooling procedure reduces the dimensionality of the input data. It replaces the given input with a local summary from the previous operation. This procedure comprises different implementations, the max pooling, and the average pooling. The max pooling summarizes by calculating the maximum value from a given region of the input, while the average pooling computes the average of this region. Figure 4 shows a practical example of both, convolution and pooling operations.

During the training procedure, the optimization algorithm adjusts the kernels and biases, employing the loss functions and algorithms presented in the previous section for fully connected networks in a very similar way. Figure 5 presents an example

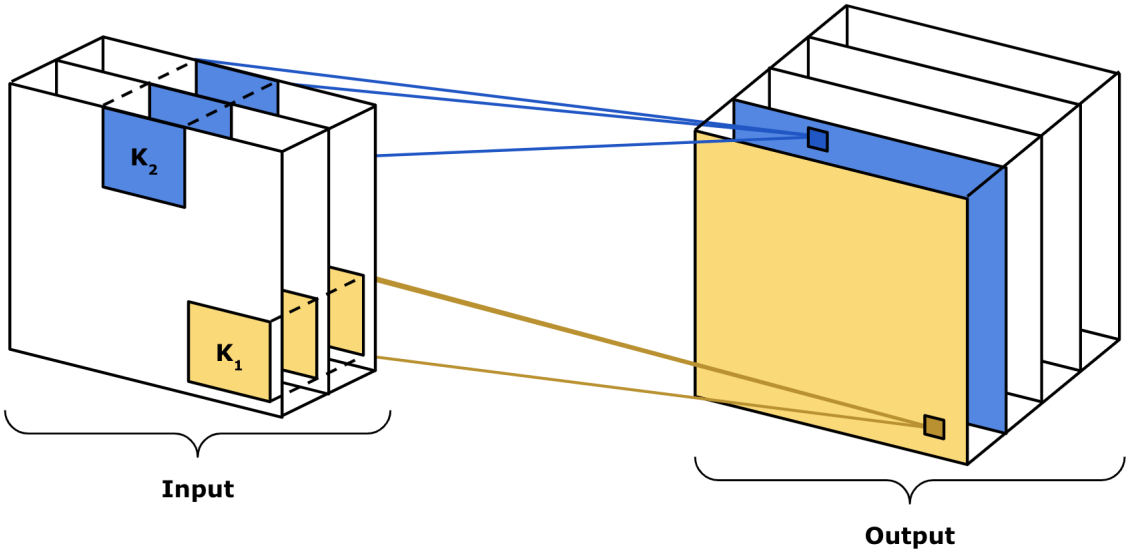


Figure 6 – Convolutional layer process representation. For each Kernel, K_i, from the discrete convolution operation, a different feature map is created as output. The outputs represented in white refer to kernels that we did not represent for simplification.

of a CNN architecture. The presented representation includes fully connected layers because of their labeling function, required to construct classifier models. Figure 6 shows feature mapping process inside convolutional layers.

2.3 ADVERSARIAL EXAMPLES

Regardless of the success of state-of-the-art convolutional neural networks, it is possible to fool these classifiers with small perturbations to their inputs. For instance, observe Figure 7. There, we applied noise to a given input initially classified as a rabbit by a CNN. As the applied perturbation is almost imperceptible, we expect that the output classification should remain the same for the crafted sample. However, as we can see, the CNN changes the classification of the output to a dog. This phenomenon reflects the lack o robustness of neural networks regarding small perturbations (SZEGEDY; ZAREMBA, *et al.*, 2013). The images designed in this way are the so-called adversarial examples.

Adversarial images raised a concern on the usage of artificial neural networks in safety-sensitive applications (i.e., with life-threatening and financial or environmental implications). However, while analyzing them in a real-world environment, some questions about their relevance raised (LU; SIBAI, *et al.*, 2017). On the other hand, further experiments showed that it is possible to create 3D models of adversarial examples that fool neural network classifiers regarding different points of view (ATHALYE *et al.*, 2017). These experiments confirmed the necessity of a deeper understanding of adversarial examples, as they consist of a real threat for critical systems that employ these technologies.

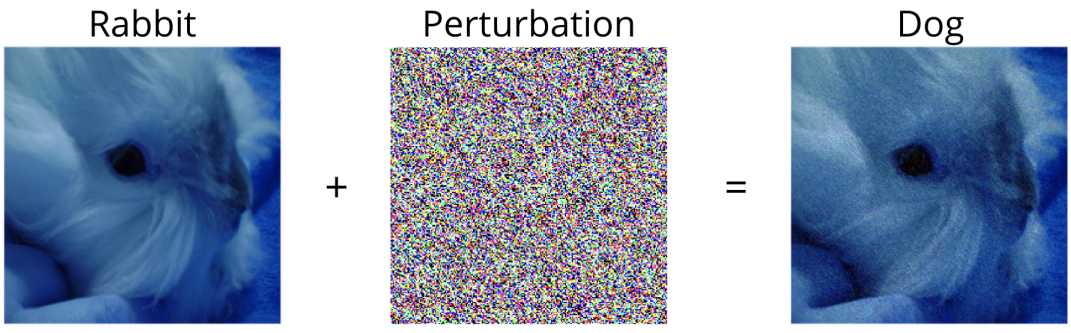


Figure 7 – Representation of an adversarial example design process, where an input image combined with a small designed perturbation generated an output image with a different classification associated by the classifier. Although, the label associated by a human remains unchanged.

2.4 ADVERSARIAL ATTACKS

Adversarial attacks are procedures designed to fool neural network classifiers. Despite the non-linearity of neural networks, these attack algorithms sight the smallest possible perturbation that leads to a change in the classification of the crafted sample (SZEGEDY; ZAREMBA, *et al.*, 2013; GOODFELLOW; SHLENS; SZEGEDY, 2014; PAPERNOT; MCDANIEL; JHA, *et al.*, 2015; MOOSAVI-DEZFOOLI; FAWZI; FROSSARD, 2016; CARLINI; WAGNER, 2017b; MADRY *et al.*, 2017). More specifically, they identify those pixels that are critical to the classifier's decision. Then subsequently, perturb the original input strategically to change its classification.

The adversarial images designed by these attacks possess the intriguing property of transferability. This characteristic is associated with the capacity of adversarial examples to transfer between different models. For instance, consider two different neural networks N_1 and N_2 created to classify the same type of inputs. The adversarial examples from N_1 are likely to transfer to N_2 in case: 1) they have different architectures; or 2) the training process occurs with disjoint data set (SZEGEDY; ZAREMBA, *et al.*, 2013; GOODFELLOW; SHLENS; SZEGEDY, 2014).

Adversarial attacks consider different threat models to design the perturbed examples. Some algorithms create input samples that a human classifies correctly, though the classifiers cannot identify the correct classification (false negative). Or examples that a human classifies incorrectly, though the CNN does so with high confidence (false positives).

We can classify adversarial attacks in two different groups according to the knowledge used to generate the perturbation: 1) the white-box attacks, which use internal information from the neural network (i.e., the number of layers, the architecture, the synaptic weights, hyperparameters, the activation functions, and training data, for instance); and 2) the black-box attacks that consider the classifier as an oracle, accessing only the output probabilities of a given input.

In general, one employs the $||.||_p - norm$ to calculate the designed perturbation magnitude. The most commonly used norms are the $||.||_0 - norm$, $||.||_2 - norm$ and $||.||_\infty - norm$. The norm $||.||_0 - norm$ measures the number of inputs changed by the perturbation, the $||.||_2 - norm$ is the euclidean distance, calculating the length of a line between the original and the perturbed input. The norm $||.||_\infty - norm$ evaluates the maximum change among all inputs.

$$||x||_p = \left(\sum_{i=1}^{n} x_i^p \right)^{\frac{1}{p}} \tag{7}$$

It is advisable to employ benchmarks for the data sets and the CNN architecture to generate comparable results while conducting experiments. For training and testing models, commonly used data sets are the:

- MNIST: data set containing 70,000 grayscale low resolution images for digits from 0 to 9 (LECUN; CORTES, 2010);
- CIFAR10: data set composed 60,000 low resolution RGB images from 10 different classes (KRIZHEVSKY; HINTON, 2012);
- ImageNet: data set comprising approximatelly 1,000,000 RGB images from 1,000 different classes (DENG *et al.*, 2009).

For the architecture of the classifier, the most used models are: LeNet (LECUN; BOTTOU, *et al.*, 1998), AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), VGG (SIMONYAN; ZISSERMAN, 2014), GoogLeNet (SZEGEDY; LIU, *et al.*, 2015), Inception (SZEGEDY; VANHOUCKE, *et al.*, 2016) and ResNet (SZEGEDY; IOFFE, *et al.*, 2017). The development and design of each of these architectures focus on specific data sets.

In the following sections, we provide an overview of different adversarial attack algorithms. We divided these attacks according to their knowledge from the classifier they aim to defeat. We started by describing the white-box attacks and the existing black-box algorithms. Among these adversarial attacks, we present those employed during the experiments proposed in this research.

White-box

We begin presenting those attacks that use at least one internal information from the classifier to design the smallest possible perturbation for a given input sample.

L-BFGS

The first adversarial attack algorithm against deep neural networks solved the problem given by the Equation 8:

$$\begin{aligned} &\min_{x'} c||\epsilon||_2 + J(x', l') \\ &s.t.\ x' \in [0,1]^n \end{aligned} \tag{8}$$

where x' represents the perturbed image, ϵ denotes the perturbation, J is the loss function, l' the target label for the input x'. And, c is the constant obtained by the solution, using a search algorithm, of the problem given by the Equation 8.

This method aims to design a perturbation for the given input sample with the smallest possible $||.||_2$-norm, such that the classification of the crafted example changes for the given target. The algorithm employed to solve the optimization problem above is the L-BFGS (SZEGEDY; ZAREMBA, *et al.*, 2013).

Fast Gradient Sign Method

The Fast Gradient Sign Method (FGSM) adversarial attack (GOODFELLOW; SHLENS; SZEGEDY, 2014) was designed to generate perturbations in a cheap one-step iterative process. As a white-box attack, this algorithm necessitates access to internal information from the classier or the training data set. The author created this method based on the hypothesis that neural networks have a linear behavior. They claimed that this linear behavior suggests the existence of these adversarial examples.

One computes the perturbation by applying the gradient of the loss function $J(\theta, x, y)$ usually the same employed for training the neural network classifier, where x is a given input image and y the target output; θ represents the weights and biases of a neural net. This attack calculates the gradient of the loss function concerning the input image instead of the trainable classifiers' trainable parameters. Then, the adversarial perturbation for each sample x is computed by Equation 9:

$$x^* = x + \epsilon \; sign(\nabla_x J(\theta, x, y)) \tag{9}$$

where ϵ is the magnitude of the perturbation and x^* is the new image which was perturbed using only the sign of the gradient in a direction to maximize the loss function (i.e., this attack does not have a target class for the perturbations). The method employed the error back-propagation algorithm to efficiently calculate the gradient of the loss function (GOODFELLOW; SHLENS; SZEGEDY, 2014).

Basic Iterative Method and Iterative Least Likely Class Method

Using some concepts of the FGSM algorithm, the authors developed some different attack approaches: the Basic Iterative Method (BIM) and the Iterative Least Likely Class Method (ILLC) (KURAKIN; GOODFELLOW; BENGIO, 2016a).

The first method propose the iterative application of the gradient sign in the direction to increase the loss function (i.e., untargeted attack), meaning that at each iteration the crafted examples will be closer to a different class, given by the Equation 10:

$$\begin{aligned} x_0 &= x \\ x_{n+1} &= Clip_{x,\xi}\left\{x_n + \eta sign(\nabla_x J(x_n, l))\right\} \end{aligned} \tag{10}$$

where η is the iteration step-size, x_i represents the input at the i-th iteration and $Clip : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}^n$ a function that limits the perturbation on the input x at each step by ξ.

The second method (ILLC) computes the gradient sign in the direction of reducing the loss function for the least likely class, according to the output of the neural

network, generating small perturbation in a iterative way given by the Equation 11:

$$\begin{aligned} x_0 &= x \\ x_{n+1} &= Clip_{x,\xi}\left\{x_n - \eta sign(\nabla_x J(x_n, l_{LL}))\right\} \end{aligned} \tag{11}$$

where l_{LL} denotes the label of the least likely class.

Jacobian-Based Saliency Map Attack

The Jacobian-Based Saliency Map Attack (JSMA) proposes a different approach for generating adversarial examples. This method employs saliency maps to find small portions of the input sample that, if slightly perturbed, leads to a significant change in the classifiers' output probabilities. This algorithm computes the forward derivative of the neural network, which corresponds to the Jacobian function of the function that represents the features learned by the neural network. Then, the saliency map is built to identify the input features to be perturbed:

$$S_i(x,t) = \begin{cases} 0, & \text{if } \frac{\partial F_t(x)}{\partial x_i} < 0 \text{ or } \sum_{j \neq t} \frac{\partial F_j(x)}{\partial x_i} > 0 \\ \left(\frac{\partial F_t(x)}{\partial x_i}\right) \left|\sum_{j \neq t} \frac{\partial F_j(x)}{\partial x_i}\right|, & \text{otherwise} \end{cases} \tag{12}$$

where $S(x,t)$ is the saliency map, $F(x)$ represents the output probabilities of the neural network classifier, t the target label for the adversarial example, i the input index and j the output index.

Finally, the algorithm applies a proper perturbation regarding the calculated saliency map. This algorithm allows crafting small adversarial perturbations that will fool the neural network with high confidence. It has a target label, making it possible to choose the desired class for the given input. However, this algorithm has a high computational cost by the necessity of computing the Jacobian of the neural network (PAPERNOT; MCDANIEL; JHA, *et al.*, 2015).

DeepFool

The DeepFool algorithm searches for an approximation to the minimum perturbation necessary to change the classification of a given input sample. Due to the non-linearity of neural networks, it is impossible to guarantee that the algorithm will return the optimal solution. This algorithm approximates the output regions for a given label as a polyhedron and employs this approximation to evaluate the distance between the current sample and the decision boundary (MOOSAVI-DEZFOOLI; FAWZI; FROSSARD, 2016).

The optimization problem solved by DeepFool is given by the Equation 13:

$$\begin{aligned} &\min_{\eta_i} ||\eta_i||_2 \\ &s.t.\ F(x_i) + \nabla F(x_i)^T \eta_i = 0 \end{aligned} \tag{13}$$

where η represents the perturbation and $F(x)$ the output probabilities of the neural network classifier.

C&W's Attack

The C&W algorithm is an attempt to defeat the Defensive Distillation method (PAPERNOT; MCDANIEL; WU, *et al.*, 2016), described further in Section 2.5. This attack aims to minimize the perturbation for a given input employing different objective functions f, and applying a variety of distance metrics (l_0, l_2 and l_∞) (CARLINI; WAGNER, 2017b). The optimization problem is given by the Equation 14:

$$\begin{aligned} &\min_{\eta} ||\eta||_p + c \times f(x+\eta) \\ &s.t.\ x+\eta \in [0,1]^n \end{aligned} \tag{14}$$

where η is the perturbation value and f the loss function.

The authors show that this adversarial attack procedure defeats a diversity of detection method, and compared to existing white-box methods, this algorithm employed fewer perturbation (CARLINI; WAGNER, 2017a)

Universal Perturbation

Instead of generating a specific perturbation for each input image, the Universal Perturbation method proposes a unique disturbance for all the samples of a data set, or in other words, a single η is computed and applied to all the attacked images. The attack algorithm utilizes a subset of inputs to reduce the computational cost of the designing process.

The universal perturbation fools different neural networks with distinct architectures due to the transferability capacity of adversarial examples. It also fools classifiers trained with disjoint data sets (MOOSAVI-DEZFOOLI; FAWZI; FAWZI, *et al.*, 2017).

Feature Adversary

Another approach comprises the design of perturbation regarding information from internal layers of the neural network classifier. Instead of minimizing the interval between the target and the original output probability, this method seeks to shrink the distance between internal feature layers. Equation 15 presents the optimization problem:

$$\begin{aligned} &arg\min_{x^*} ||Z_k(x^*) - Z_k(x)||_2^2 \\ &s.t. ||x^* - x||_\infty < \delta \end{aligned} \tag{15}$$

where Z_k represents the output of the k-th layer.

This method relies on the claim that adversarial images are too close to the original examples in the pixel space, although their internal representation bears a resemblance from images that belong to the misclassified label (SABOUR *et al.*, 2015).

Projected Gradient Descent

The Projected Gradient Descent (PGD) method (MADRY *et al.*, 2017) aims to craft perturbations for an input image in an iterative process to generate adversarial examples with a small disturbance compared to other existing attack algorithms. This method is considered the universal first-order attack algorithm.

The PGD adversarial attack algorithm is essentially a multi-step iterative variation of the FGSM. As a white-box attack, it also uses internal information from the neural network classifier to compute the gradient of the loss function regarding a given input image. The error back-propagation algorithm calculates the gradients. Equation 16 presents the iterative calculation of the perturbation :

$$x_{i+1} = x_i + \epsilon \; sign(\nabla_x J(\theta, x, y)) \tag{16}$$

where $J(\theta, x, y)$ corresponds to the loss function, usually the same employed during the training procedure, x is a given image, y the target output, θ the weights and biases of a neural network, ϵ controls the magnitude of the perturbation designed at each step of the attack and i the iteration of the algorithm.

The PGD attack can be employed in two different ways, the $||.||_2 - norm$ and the $||.||_\infty - norm$. In Equation 16 we have the $||.||_\infty - norm$ representation. The iterative calculation for the $||.||_2 - norm$ attack is presented in Equation 17:

$$x_{i+1} = x_i + \epsilon \; \frac{\nabla_x J(\theta, x, y)}{||\nabla_x J(\theta, x, y)||_2} \tag{17}$$

Black-box

In this section, we describe those attacks that use the neural network classifier as an oracle.

Substitute Model

The transferability of adversarial examples provides the possibility of designing perturbations such that internal information from the classifier is not required. The main idea is to create a substitute model that approximates the decision boundaries of the original classifier. To acquire this approximation, the attacker gives artificially created samples and collects its output probability distribution. Posteriorly, the algorithm creates a training set for training a substitute model.

It is possible to generate adversarial examples for the substitute model that will possibly transfer to the original classifier with any white-box attack. This property

permits using this procedure for different classifiers other than neural networks (PAPERNOT; MCDANIEL; GOODFELLOW, *et al.*, 2017).

This approach defeats defenses that use gradient masking because this algorithm uniquely requires the classifiers output probabilities. However, due to the necessity of training a different neural network classifier, this approach can be computationally expensive.

Local Search Attack

The Local Search Attack (LSA) proposed a greedy local-search algorithm to generate adversarial examples in a black-box condition. This approach randomly selects pixels of a given input sample, generates new images with one of the selected pixels perturbed, according to Equation 18 (NARODYTSKA; KASIVISWANATHAN, 2016).

$$x_{pert}^{a,b}(u,v,c) = \begin{cases} x(u,v,c), & \text{if } a \neq u \text{ or } b \neq v \\ p \times sign(x(u,v,c)), & \text{otherwise} \end{cases} \tag{18}$$

Subsequently, the algorithm selects T pixels, from those selected previously, that generated the images that mostly minimized the proposed loss function $f_{l(x)}(x) = F_{l(x)}(x)$, and generate a new image with all the T pixels perturbed, according to Equation 19.

$$x_{adv}(u,v,c) = \begin{cases} r \times x(u,v,c) + (UB - LB), & \text{if } r \times x(u,v,c) < LB \\ r \times x(u,v,c) - (UB - LB), & \text{if } r \times x(u,v,c) > LB \\ r \times x(u,v,c), & \text{otherwise} \end{cases} \tag{19}$$

If the generated image is an adversarial example or the algorithm reaches the maximum number of iterations, the process stops. Otherwise, it selects new pixels among the neighbors within a distance s from the previously chosen pixels and repeats the procedure from the beginning.

Zeroth Order Optimization

Given a neural network N and an input image x_0 that belongs to the class l_0, the Zeroth Order Optimization Method (ZOO) computes the adversarial x^* associated to x_0 by solving the following optimization problem:

$$\min_x g(x) = \min_x(||x - x_0||_2^2 + c \times f(x)) \tag{20}$$

$$s.t.\ x \in [0,1]^n$$

$$f(x) = \max\{log[F_{l_0}(x)] - \max_{i \neq l_0}\{log[F_i(x)\}], -\kappa\}, \tag{21}$$

where $f(x)$ is a function that depends on the difference between the two highest confidence scores in $F(x)$, c and κ are positive constants, and $||x - x_0||_2^2$ is a normalization factor used to obtain adversarial images x^* as close as possible to x_0.

According to Equation (20), the adversarial x^* is an image similar to x_0 that makes the neural network change its classification output from l_0 to another class. To solve the optimization problem above, the ZOO attack approximates the gradient of $F(x)$ with respect to the coordinates x_i by the quotient:

$$\frac{\partial f(x)}{\partial x_i} \approx \frac{f(x + h \times e_i) - f(x - h \times e_i)}{2h}, \tag{22}$$

where h is a small and fixed constant and e_i is the standard basis vector with the i-th component set to one and all other components equal to zero.

Finally, the increment added at each iteration of the optimization algorithm to obtain the image x_i^{k+1} from x_i^k is modulated by a factor η, e.g., if the Newton's method is used to solve Equation (20), then $x_i^{k+1} = x_i^k + \eta \times \frac{\partial f}{\partial x_i} \Big/ \frac{\partial f^2}{\partial x_i^2}$, instead of simply $x_i^{k+1} = x_i^k + \frac{\partial f}{\partial x_i} \Big/ \frac{\partial f^2}{\partial x_i^2}$. In this study, we employed the ADAM optimizer to solve the optimization problem presented in Equation 20.

One Pixel Attack

To generate adversarial examples with fewer perturbations regarding the $||.||_0$-norm, or in other words, by changing a slight amount of pixels, the One pixel attack algorithm was designed (SU; VARGAS; SAKURAI, 2017).

This attack generates adversarial examples by perturbing only one pixel from the input sample ($||\eta||_0 = 1$). By means of the differential evolution method, from the evolutionary computation, this attack aims to find those pixels that are critical for the assignment of the input image with a given class. Equation 23 shows the problem formulation:

$$\begin{aligned} &\min_{\eta} -F_t(x + \eta) \\ &s.t. ||\eta||_0 = 1 \end{aligned} \tag{23}$$

where η is the perturbation, $F(x)$ the output probabilities of the neural network classifier, t the target class and d the maximum perturbed pixels.

Expectation Over Transformation

The Expectation of Transformation (EOT) attack answers the doubt raised above the relevance of adversarial examples for real-world applications (LU; SIBAI, *et al.*, 2017). More specifically, this algorithm presents the possibility of creating adversarial examples that maintain the misclassification over different viewpoints.

This method crafts adversarial examples using defined transformations, representing changes in the points of view (i.e., translation, rotation, or scaling) applied to the input sample. It enabled the development of adversarial images (2D) and objects using a 3D printer that could fool artificial neural networks regarding different points of view (ATHALYE *et al.*, 2017).

Adversarial Patch

Traditionally adversarial attack algorithms aim to craft slight perturbations so that the distance between the clean and the designed examples is as smaller as possible. The Adversarial Patch attack creates adversarial perturbations that are not imperceptible and do not change a human classification. The patch development process takes under consideration a subset of transformations (i.e., scaling or rotation) and the applied location.

The Adversarial Patch attack generates a perturbation that can be printed or applied to the image virtually and can fool the classifier by changing the classification to a target class defined in the generation of such perturbation (BROWN *et al.*, 2017). This approach employs a variant of the Expectation Over Transformation algorithm during the determination of the patch (ATHALYE *et al.*, 2017).

Natural GAN

The Natural GAN attack algorithm proposes a different approach to generate the adversarial perturbation. This method operates with a Generative Adversarial Network (GAN) trained on a data set to generate images representing the training data. Following, the authors propose the use of an Inverter which is a function that maps input images to the latent space of the generator from the GAN. This function enables the applications of the perturbation on the latent variables of an input image.

Perturbing latent variables allow the creation of different output images that carry features from the trained data set to the newly crafted example. Searching in the latent space for adversarial example also qualifies adversaries to be legible images and semantically similar to the original input (ZHAO; DUA; SINGH, 2018).

Semantic Adversarial Examples

One proposes an adversarial attack algorithm that generates perturbed examples by changing the colors of the given input. This method relies on the human vision bias over shapes rather than by colors. The problem of creating semantic adversarial examples can be described by Equation 24:

$$\begin{aligned} &find\ x^* \\ &s.t.\ \Omega(x^*) = \Omega(x)\ and\ F(x^*) \neq F(x) \end{aligned} \tag{24}$$

where Ω stands for the human vision system, F is the neural network classification, x the original input and x^* the adversarial example.

The algorithm consists of an input image transformation from RGB to HSV (Hue, Saturation, and Value). This conversion allows the application of changes to the values of Hue and Saturation. The Value guarantees that the shape of the input sample remains the same by transforming singularly the colors (HOSSEINI; POOVENDRAN, 2018).

2.5 ADVERSARIAL DEFENSES

Adversarial defense methods are tools designed to overcome attack algorithms. These methods employ different approaches and focus on distinct objectives: increase the robustness of the classifier regarding small perturbations, identify or reconstruct adversarial examples, formal verification of the classifiers functioning, for instance.

At this point, we go through five different existing approaches to deal with these malicious examples: adversarial training, defensive distillation, adversarial detection, input reconstruction, and network verification. For each of them, we present several works from the literature and their achievements.

Adversarial Training

The first proposed approach to increase the robustness of neural networks against malicious examples was the adversarial training, which comprises augmenting the training data set with perturbed inputs generated using one or more adversarial attacks (SZEGEDY; ZAREMBA, *et al.*, 2013).

There already exist two distinct ways of implementing the adversarial training method: a) in the first one, adversarial examples are generated during the training procedure, iteratively and within a maximum perturbation value, until the classifier learns to classify these examples; b) the second approach comprises training a neural network with the original training set, then generate adversarial examples using some of the malicious attacks to augment the training set for optimizing a second neural network classifier (NA; KO; MUKHOPADHYAY, 2018).

Empirical tests showed that adversarial training increased the robustness of neural networks against one-step attacks and that iterative attacks are less transferable than one-step attacks. Training neural networks with a one-step attack is also an alternative for defending against black-box attacks, which is related to the decrement of the transferability (KURAKIN; GOODFELLOW; BENGIO, 2016b).

By applying the adversarial training with the PGD attack (reliable first-order malicious attack), using the approach a), for a neural network trained on the MNIST data set, the CNN achieved an accuracy of at least 89% against stronger adversaries,

being robust even against iterative white-box attacks. For the CIFAR10 data set, the accuracy reached 46% against the same adversaries (MADRY *et al.*, 2017).

Instead of optimizing the neural network with adversarial examples generated by a single attack, the authors proposed creating and training auxiliary models, attack these classifiers with a diversity of different methods and augment the training set with all of these generated adversarial examples. With the augmented training set, one can optimize another neural network classifier to increase its robustness (TRAMÈR *et al.*, 2018).

Theoretical results showed that comparing a neural network and the k-nearest neighbors (KNN) model, the second is sample efficient regarding the first one. In other words, it means that the amount of necessary data to make a neural network robust to perturbation, of size ϵ, by using adversarial training, is higher compared to KNN, and it grows according to the codimension (the difference between the input dimension and the data manifold dimension) of the data set (KHOURY; HADFIELD-MENELL, 2019).

Defensive Distillation

The network distillation method consists of a simple approach to reduce the computational cost of neural networks. The principal reason was to apply neural network models in devices with lower computational capacity, such as smartphones (HINTON; VINYALS; DEAN, 2015). This method comprises training two neural networks. The first neural network uses the training data set and desired outputs (i.e., hard labels) during the training process, with a higher temperature T in the softmax activation function, given by Equation 25. Hard labels are output probabilities with 100% assigned to the correct classification and 0 to all the other classes.

$$F(X) = \left[\frac{e^{\frac{z_i(X)}{T}}}{\sum_{l=0}^{N-1} e^{\frac{z_l(X)}{T}}} \right] \tag{25}$$

The method proposes training the second neural network, which has a more compact architecture, with the same input samples employed to the first model, except that the desired labels are the outputs of the first classifier regarding the input samples applied during the training process, denoted as soft labels. The temperature remains the same for the second neural network training.

Defensive distillation algorithms make use of the network distillation method, except that this method does not reduce the neural network architecture for the second trained model, as this approach seeks to increase the robustness of neural networks instead of reducing the computational cost (PAPERNOT; MCDANIEL; WU, *et al.*, 2016).

The authors claim that the proposed method increases the robustness of the second neural network model, as it enlarges the smoothness of this model with the increment on the temperature employed in the softmax function. There is an increment

in the amount of information provided for training the second neural network, as some classes share features captured by the soft labels.

Besides the success of defensive distillation against some adversarial attacks, other algorithms circumvented such a defense method, for example, C&W (CARLINI; WAGNER, 2017b), and the substitute model (PAPERNOT; MCDANIEL; GOODFELLOW, *et al.*, 2017).

Adversarial Detection

The detection of adversarial examples comprises another approach employed to circumvent the flaw in the robustness of neural networks regarding these malicious examples. Differently from previously presented methods, the detection was studied jointly with a range of distinct techniques. These approaches aim to identify instead of reduce existing adversarial examples.

One of these approaches employed deep neural networks as a binary classifier, trained on generated adversarial examples, to identify if the given inputs are clean or adversarial examples (METZEN *et al.*, 2017). Just in case of being classified as an ordinary example, the model feeds into the classifier. Another approach incremented a new class in the outputs of the original classifier, denoted as the outlier class, where this extra label is responsible for identifying if the input is an adversarial example (GROSSE *et al.*, 2017).

Employing Radial Basis Function Support Vector Machine, SafetyNet gets as input the outputs of each layer from the deep neural network and uses it to identify if the entry is an adversarial example. The authors claimed that it is hard to defeat their proposed algorithm because it is necessary to fool both the neural network and also the detector (LU; ISSARANON; FORSYTH, 2017).

With the claim that the uncertainty of adversarial examples is higher than the on clean data, one proposed a Bayesian neural network to measure this uncertainty and classify this input as an original or adversarial example (FEINMAN *et al.*, 2017).

The discovery that the adversarial examples distribution differs from the clean data distribution made it possible to develop a new detection algorithm. This approach detected perturbed examples created with FGSM, BIM, DeepFool, and C&W (SONG *et al.*, 2017).

Despite the success in the identification of some adversarial examples generated by different attack algorithms, it was possible to defeat all of these detection methods by attacking them with the C&W algorithm. It was possible with the application of some changes in the loss function. These results showed that the process of detecting adversarial examples is not a simple task (CARLINI; WAGNER, 2017a).

Input Reconstruction

Another proposed way of defending against adversarial examples relies on reconstructing the inputs to transform the given malicious inputs so that the generated samples do not fool the classifier. This approach reduces the number of adversarial examples instead of increasing the classifier's robustness regarding these attacks.

Some of these approaches used autoencoders to reach the original example or retrieve the original classification (GU; RIGAZIO, 2014). Another use of autoencoders remained on reconstructing these adversaries back to the distribution of clean inputs (SONG *et al.*, 2017).

MagNet algorithm reconstructs adversarial examples by primarily applying a Gaussian noise to the input sample, and as a second plan, employs an autoencoder to retrieve the original classification associated with the given input (MENG; CHEN, 2017).

Network Verification

Although the network verification problem is provably NP-complete (KATZ *et al.*, 2017), it is still a promising solution to defend against adversarial examples because it allows the detection of unknown counterexamples and the previous identification of flaws in the classification procedure.

There are three main classes to divide these algorithms according to the problem each of them aims to solve (LIU *et al.*, 2019). The first set of algorithms returns counterexamples, which are examples classified with a different label regarding the original class (HUANG *et al.*, 2017).

Denoted as adversarial result, the second aims to find and return the maximum allowed perturbation that, if applied to a given original input, will not cause a transformation on the output classification (TJENG; XIAO; TEDRAKE, 2017).

Finally, the third class of algorithms, which focuses on computing the reachable set for some inputs of a given label and compares it to the expectable output set, based on the problem constraints and the other output set of the other classes (XIANG; TRAN; JOHNSON, 2017).

Proposed Methods

Different from adversarial training, defensive distillation, and network verification, which aim to reduce the amount of existing perturbed examples, in this research, we propose two different approaches that focus on reducing the impact of adversarial examples in the classification tasks of CNN models, decreasing the efficacy of adversarial attacks.

We classify the first proposed approach as a gradient masking method, similarly to adversarial training and defensive distillation (PAPERNOT; MCDANIEL; GOODFEL-

LOW, *et al.*, 2017) because it inserts a controlled disturbance (noise) to the outputs of a neural network classifier, making the correct direction to design a perturbation uncertain for a black-box attack algorithm that uses these output probabilities (uncertain gradient).

Due to the way this method works, it is possible to combine it with different defensive methods, including those that increase the classifier robustness (MADRY *et al.*, 2017) and adversarial detectors (SONG *et al.*, 2017), as the proposed method also forces the attacker to increase the perturbation inserted to the input images.

The second proposed method enables developing a new detector using a different feature compared to existing detection methods. According to our results for this approach, the first digit distribution of adversarial examples generated by some white-box attacks diverges from the first digit distribution associated with clean inputs.

Our results also suggest that, by the behavior of the divergence of the empirical and the theoretical first digit distribution regarding the attack iterations, it is possible to identify neural network classifiers under attack for several cases.

3 DISTURBANCE-BASED DEFENSE

In this chapter, we propose a method to defend neural networks against black-box adversarial attacks. This method relies on the perturbation of the output layer of a network.

3.1 PROBLEM FORMULATION

Let N be any convolutional neural network for image classification and A be a black-box adversarial attack. We shall use $A(N, I_0)$ to denote the set of adversarial images generated by A for N, regarding a subset of seed images I_0. $F(x) \in \mathbb{R}^m$ represent the m-dimensional output of the softmax layer of N when presented with an n-dimensional image vector $x \in [0,1]^n$. N classifies this input x into one of the m available classes. We assume that some coding mechanism is available to convert matrices representing images to column vectors.

Let us now call N' the network that is equal to N, except for the output of its softmax layer, which is now given by $G(F(x) + \mathbf{d})$, where $\mathbf{d} \in \mathbb{R}^m$ is a controlled disturbance vector, and $G : \mathbb{R}^m \to \mathbb{R}^m$ is a normalization function, given by Equation (26). G makes the outputs of the network N' sum up to one, retaining the form of a probability distribution over the classes $i \in \{0, \ldots, m-1\}$.

$$G(F(x) + \mathbf{d})_i = \frac{(F(x) + \mathbf{d})_i}{\sum_{j=0}^{m-1} (F(x) + \mathbf{d})_j} \tag{26}$$

Our goal is to design such $\mathbf{d}$ in a way that:

1. the success rate of N' on the classification of $A(N', I_0)$ is higher than that of N for $A(N, I_0)$; and

2. for any $i, j \in \{0, \ldots, m-1\}$, if $F_i(x) \leq F_j(x)$, then $d_i + F_i(x) \leq d_j + F_j(x)$. That is, the disturbance vector $\mathbf{d}$ does not affect the ordering of the values of $F(x)$; and

3. for any $i \in \{0, \ldots, m-1\}$, the inequality $|d_i| \leq \delta \times F_i(x)$ holds, where δ is a design parameter satisfying $0 < \delta < 1$. In other words, it limits the designed disturbance to be within the closed interval $[-\delta \times F_i(x), \delta \times F_i(x)]$.

3.2 DEFENSE PROPOSALS

We selected a black-box setting as a constraint for applying this defense method, as it is a much more realistic scenario for a malicious attack. The proposed approach aims to hinder the input critical pixels identification by the attacker. The application of a controlled disturbance to the output probabilities complicates the approximation of the

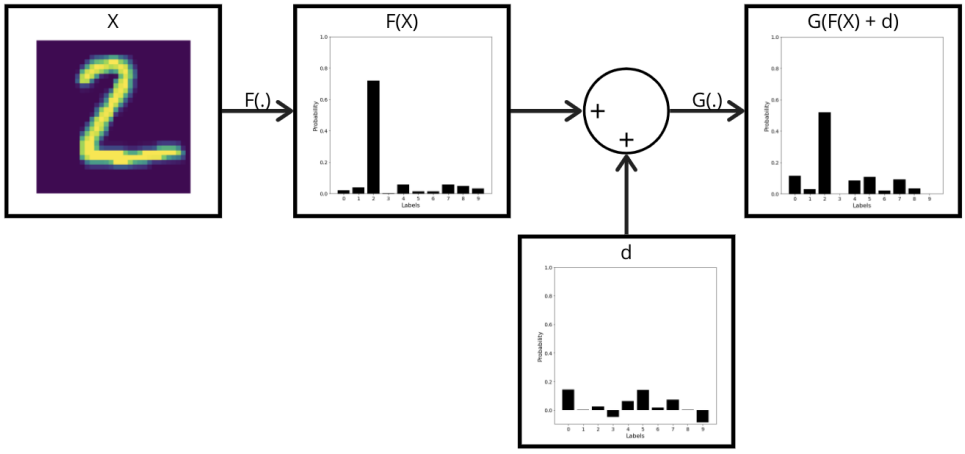


Figure 8 – Overview visualization of the proposed approaches, composed of three steps: a) class probability distribution calculation in a black-box setting, represented by $F(.)$; b) controlled disturbance addition, , to the output probabilities; and c) normalization of $F(X) + d$, represented by the mapping $G(.)$.

gradient regarding a given input. One possible application for the proposed method is to defend commonly used online deep learning API.

We describe in this section three different approaches to design a disturbance vector $\mathbf{d}$ for adding to the output layer of a neural network. Figure 8 presents an overview visualization of the proposed defense method, where we see that this method intends to reduce the amount of knowledge obtained by the attacker during the inference process, compared to the black-box setting.

The statement (1), previously presented in the problem formulation, constitute the objective of the proposed methods. The only one that satisfies the requirements (2) and (3) is the last method, termed Limited Disturbed Classes with Order Preservation (Subsection 3.2.3). We present the other two, though, as a means to later evaluate the performance of our ideas when we relax the constraints on $\mathbf{d}$ posed by the problem formulation.

3.2.1 Homogeneously Disturbed Classes (HDC)

This first method generates a scalar disturbance $\mathbf{d}$ (or noise) that is homogeneously applied to each $F(x)_i$ using the same normal distribution $\mathcal{N}(0, \sigma_f)$ with 0 mean and standard deviation σ_f. The standard deviation σ_f was defined to be equal to $\Delta \div r$, where we choose a Δ equal to half of the difference between the largest and the second higher value of the vector $F(x)$. We divided Δ by a parameter r so that the saturation function, defined further, does not affect the disturbance's random behavior. For all the experiments presented in this paper, we employed $r = 3$.

In Algorithm 1, which describes the HDC method, the function $sort(F(x))$ returns a m-dimensional vector whose values are sorted in descending order, i.e., the largest

Algorithm 1: Homogeneously Disturbed Classes

Input: $F(x) \in \mathbb{R}^m$
$\sigma_f \leftarrow (sort_0(F(x)) - sort_1(F(x))) \div (2 \times r)$;
for $i \in \{0, \ldots, m-1\}$ **do**
  Select a random d from distribution $\mathcal{N}(0, \sigma_f)$;
  $F(x)_i \leftarrow F(x)_i + sat_{\sigma_f \times r}(d)$;
end
return $G(F(x))$;

value is the first element, represented by $sort_0(F(x))$. Additionally, $sat_z(x)$ denotes the saturation function, which is equal to x, if $|x| \leq z$, and to $z \times sign(x)$, otherwise.

3.2.2 Disturbed Classes with Order Preservation (DCOP)

In this second approach, our goal is to change the HDC method to attain the requirement (2) of the problem formulation. To simplify the notation, and without loss of generality, we assume that the vector $F(x)$ given as input to our algorithm is already sorted in descending order. Still, we define the map $diff : \mathbb{R}^m \times \{0, \ldots, m-1\} \rightarrow \mathbb{R}$ to be:

$$diff(\mathbf{v}, i) = \begin{cases} \mathbf{v}_i - \mathbf{v}_{i+1}, & i = 0 \\ \min\{\mathbf{v}_i - \mathbf{v}_{i+1}, \mathbf{v}_{i-1} - \mathbf{v}_i\}, & 1 \leq i \leq m-2 \\ \mathbf{v}_{m-2} - \mathbf{v}_{m-1}, & i = m-1 \end{cases}$$

This map computes the distance between some element i of **v** and its nearest neighbor, where **v** is the output probabilities of the classifier in descending order, that is, the standard deviation for calculating the disturbance associated with each output probability represents a fraction of the distance calculated by the $diff$ map.

The algorithm below shows that, now, the magnitude of each d_i is delimited by $diff(F(x), i)$, thus keeping the ordering of $F(x)$ intact.

Algorithm 2: Disturbed Classes with Order Preservation

Input: $F(x) \in \mathbb{R}^m$
for $i \in \{0, \ldots, m-1\}$ **do**
  $\sigma_{v,i} \leftarrow diff(F(x), i) \div (2 \times r)$;
  Select a random d_i from distribution $\mathcal{N}(0, \sigma_{v,i})$;
  $F(x)_i \leftarrow F(x)_i + sat_{\sigma_{v,i} \times r}(d_i)$;
end
return $G(F(x))$;

Although this new version does satisfy requirement (2) of the problem formulation, it does not attain specification (3). In the subsequent subsection, we deal with the last variant of our disturbance approach called Limited Disturbed Classes with Order Preservation.

3.2.3 Limited Disturbed Classes with Order Preservation (LDCOP)

In this approach, the standard deviation σ_v of $\mathcal{N}(0, \sigma_{v,i})$ is computed in the same way as in DCOP. However, $\delta \times |F(x)|$ limits the magnitude of each d_i in this variant, as presented in the pseudocode that follows. We assume that $F(x)$ is in descending sorted order.

Algorithm 3: Limited Disturbed Classed with Order Preservation

Input: $F(x) \in \mathbb{R}^m$ and $\delta \in [0, 1]$;
for $i \in \{1, \ldots, m\}$ **do**
  $\sigma_{v,i} \leftarrow diff(F(x), i) \div (2 \times r)$;
  Select a random d_i from distribution $\mathcal{N}(0, \sigma_{v,i})$;
  $F_i(x) \leftarrow F_i(x) + sat_{\delta \times F_i(x)}(sat_{\sigma_{v,i} \times r}(d_i))$;
end
return $G(F(x))$;

3.3 EXPERIMENTAL SETUP

We present in this section the three main elements used to evaluate the proposed defense method against black-box attacks: the CNN, the experimental scenarios, and the employed adversarial attack.

3.3.1 Convolution neural network (CNN)

We employ a CNN of 12 layers (described in Table 1) that makes use of convolution functions interleaved with batch normalization and max pooling operations. In Table 1, we use $\mathrm{conv(m, n)} - \mathrm{c}$ referring to a convolutional layer with kernel of size $m \times n$ and c channels. ReLU activation function was applied for all the layers, except for the last one.

The classifier model presented above (Table 1), was trained on the MNIST data set (LECUN; CORTES, 2010), for further information about this data set, refer to the Section 2.4. For training this model, we divided the MNIST data set into three different disjoint subsets: training set, validation set, and test set.

The training set comprised $50,000$ images, whereas the validation and test sets, $10,000$ samples each. ADAM (KINGMA; BA, 2014) was used as the optimization method to adjust the CNN parameters, minimizing a categorical cross-entropy loss on the training set. For such an optimization process we applied a learning rate of $5\mathrm{e}{-5}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1\mathrm{e}{-08}$. The CNN was trained for approximately 50 epochs, resulting in an accuracy of 99.59% and 97.71% on the training set and test set, respectively.

We trained a second neural network classifier over a data set comprising concentric circles, presented in Figure 9, as proposed in (KHOURY; HADFIELD-MENELL, 2019). A fully connected neural network, represented in Table 2, was trained over this data set, where the training set comprises $2,000$ samples and 500 samples for the validation and test sets.

The ADAM optimizer was employed during the training process, with learning rate of $5\text{e}{-5}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1\text{e}{-08}$. The Classifier was trained for about

Table 1 – Neural Network Architecture for the MNIST data set - Disturbance-based defense method

Layer	Type	Dimensions
0	Input	(32x32x3)
1	Conv(3x3)-32	(32x32x32)
2	Conv(3x3)-32	(32x32x32)
3	Batch Normalization	(32x32x32)
4	Max Pooling(2x2)	(16x16x32)
5	Conv(3x3)-64	(16x16x64)
6	Conv(3x3)-64	(16x16x64)
7	Batch Normalization	(16x16x64)
8	Max Pooling(2x2)	(8x8x64)
9	Fully Connected	(1024)
10	Batch Normalization	(1024)
11	Fully Connected	(1024)
12	Fully Connected	(10)

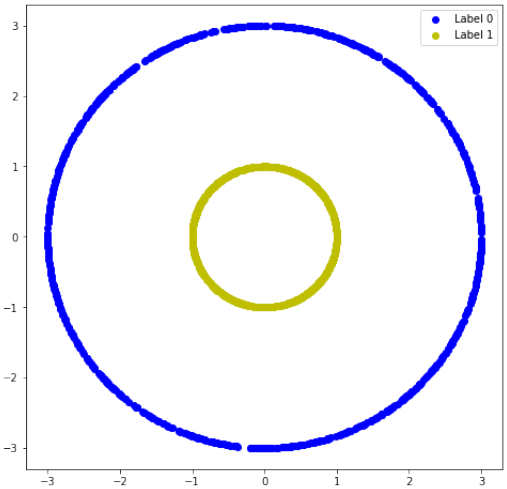


Figure 9 – Scatter plot of the concentric circles data set, employed to visualize the behaviour of the ZOO attacker against the neural network classifier, with and without the proposed defense methods. The data set consist of two concentric circumferences, with radius equal to 3 (Label 0) and 1 (Label 1).

Table 2 – Neural Network Architecture for the concentric circles data set - Disturbance-based defense method

Layer	Type	Dimensions
0	Input	(2)
1	Fully Connected	(100)
2	Fully Connected	(1)

Table 3 – Defense methods for CNN classifier

v	Method
0	Original CNN
1	HDC
2	DCOP
3	LDCOP10
4	LDCOP20
5	LDCOP50

40 epochs, reaching an accuracy of 100% on both, the training set and test set.

3.3.2 Experimental procedure

Below we describe the test scenarios used to evaluate the performance of our defense approaches:

1. Set the maximum number of iterations of the ZOO algorithm to 10;
2. Set $h = 0.0001$ and the optimizer parameters $\beta_1 = 0.9, \beta_2 = 0.99$ and $\epsilon = 1\text{e}{-8}$, as proposed by the authors of the ZOO method (CHEN *et al.*, 2017);
3. Randomly select 500 images from the training set[1];
4. For each $v \in \{0, \ldots, 5\}$, obtain the corresponding neural network N_v according to Table 3, where LDCOPX denotes the LDCOP defense strategy with $\delta = X/100$.;
5. For each $\eta \in \{0.01, 0.05, 0.1\}$, compute the set of adversarial images $S_\eta(N_v)$ using the ZOO algorithm;
6. Classify the sets of examples $S_\eta(N_v)$ with the network N_v;

[1] Preliminary experiments show equivalent results with images selected from the test set.

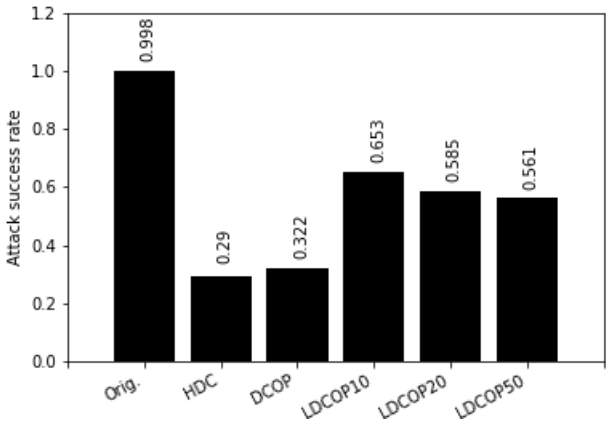


Figure 10 – Attack success rate for the ZOO algorithm against different variants of CNNs, with and without defenses. The leftmost bar shows the result for the original CNN without any defense.

3.3.3 Adversarial Attack

The ZOO algorithm (CHEN *et al.*, 2017), which is a black-box attack, was employed to generate the adversarial examples associated with the neural network classifiers presented previously for each of the variations proposed in this research. For further details on these attack algorithms, refer to Section 2.4.

3.4 RESULTS

We evaluate the two main effects of our defense method: the reduction in the attack success rate and the increment in the magnitude of the attacker's perturbation.

3.4.1 ZOO attack success rate reduction

Figure 10 shows the success rate of the adversarial images generated by ZOO for the original network as well as for the networks enhanced with our defense methods, i.e., with the disturbance $\mathbf{d}$ added to $F(x)$. The attack success rate represents the percentage of samples, from a set of attacked images, that turned into adversarial examples. We observed a significant accuracy improvement of the defended networks while classifying the candidate adversarial images. Our method reduced the attack success rate from 99.8% to 65% in the most restricted scenario (variant LDCOP10), relative to the ZOO attack. The different performances of the defense strategies are associated with the restrictions imposed on the additive disturbance $\mathbf{d}$: the more properties of $F(x)$ the disturbance $\mathbf{d}$ must preserve, the less effective is the defense. In other words, as the LDCOP approach attains properties (2) and (3) from problem formulation, we expected its performance to be lower or equal compared to the other methods, while HDC aims solely in the objective, stated in (1), without satisfying the remaining requirements.

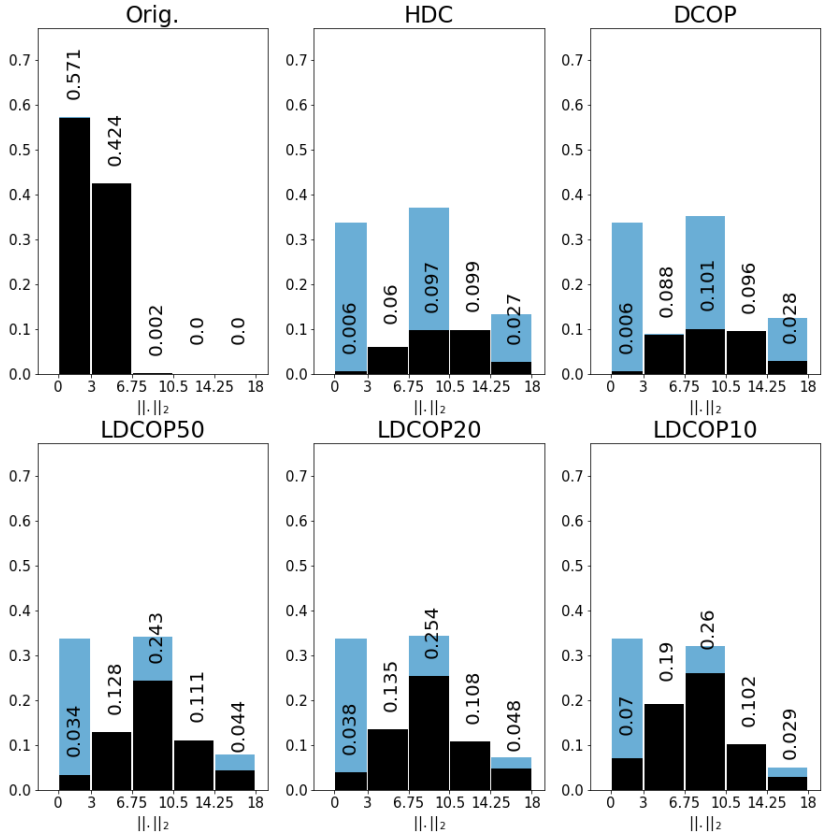


Figure 11 – Distribution of the perturbations' magnitude added by ZOO to generate adversarial images from a given set of images. We divided the x-axis into five intervals of perturbation magnitudes. The y-axis shows the relative frequency associated with each of these intervals. For example, 57.1% of the adversarial images generated for the original network has a perturbation magnitude between 0 and 3, whereas only 0.6% of all images lie in this interval for the network defended with HDC. The black bars correspond to effective adversarial examples, whereas the blue ones represent the total candidates of adversaries. The first interval is smaller than the others as the Euclidean norm is greater or equal to 0.

3.4.2 Perturbation magnitude increment

Let $I = \{x_0, x_1, \ldots, x_i\}$ be a set of images $x_k \in \mathbb{R}^n$ and $A(N_v, I) = \{x_0^{adv}, x_1^{adv}, \ldots, x_i^{adv}\}$ be the corresponding set of adversarial images generated by ZOO for the network N_v, $v \in \{0, \ldots, 5\}$. Define α_k to be $||x_k^{adv} - x_k||_2$, i.e., the 2-norm of the perturbation added to x_k in order to generate an adversarial x_k^{adv}.

Figure 11 presents the distribution of the α_k for all six network variants. We can see that the defenses forced the attack algorithm to increase the magnitude of the applied perturbation. The higher the magnitude of α_k, the less similar is x_k^{adv} to x_k, making x_k^{adv} a weaker and possibly a more questionable adversarial image. Therefore, our method not only decreases the attack success rate but also forces the creation of intensively perturbed adversarial examples (generating adversaries of worse quality).

Figure 11 also shows the distribution of those α_k associated with the images x_k

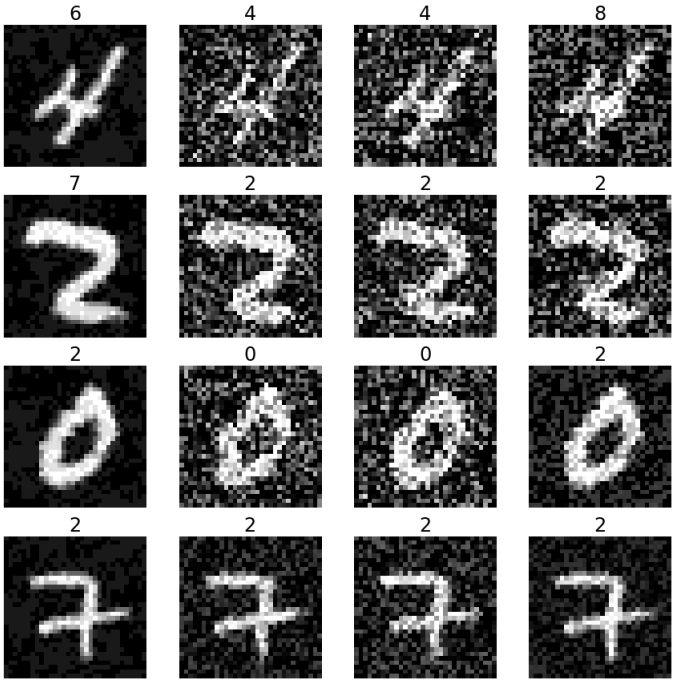


Figure 12 – Examples of images generated by ZOO to attack different CNNs trained on MNIST. Starting at the leftmost column, we have a sample of images generated for the following networks: 0 (Original CNN), 1 (HDC), 2 (DCOP), and 3 (LDCOP with $\delta = 0.1$). The class with the highest score according to the corresponding classifier is shown above each image. Note how the quality of the adversarial images worsens when a defense method is employed.

that indeed fooled the classifier (black bars). We observe that for slight perturbations ($\alpha_k \leq 3$), the percentage of adversarial examples declined from 57.1% for the original network to 7.0% for the network defended with the LDCOP10 approach, corresponding to a reduction of 87.7% in that interval.

Figure 12 depicts the quality degradation of the images generated by the ZOO algorithm. For each column, it displays candidate adversarial images and the label predicted by the respective network. Note how these images have a visually higher perturbation in columns 1, 2, and 3 (the ones where we employ our defenses) compared to column 0. This degradation is a consequence of the increased perturbation forced by the defense methods.

Figure 13 shows the gradient approximation computed using the ZOO algorithm for each proposed defense method. We can see that the gradients computed for N_v, with $v \in \{1, 2, 3\}$ (when the defense is active) have no spatial regularities through the image pixels when compared with the gradient computed for the neural network N_0 (*Orig.*). This result incurs from the induced uncertainty in the gradient approximation by disturbing the neural network's output probabilities. The attacker is left confused about which direction it should perturb the image: the uncertainties perceived by the attacker are spatial (through the image) and temporal (at each iteration).

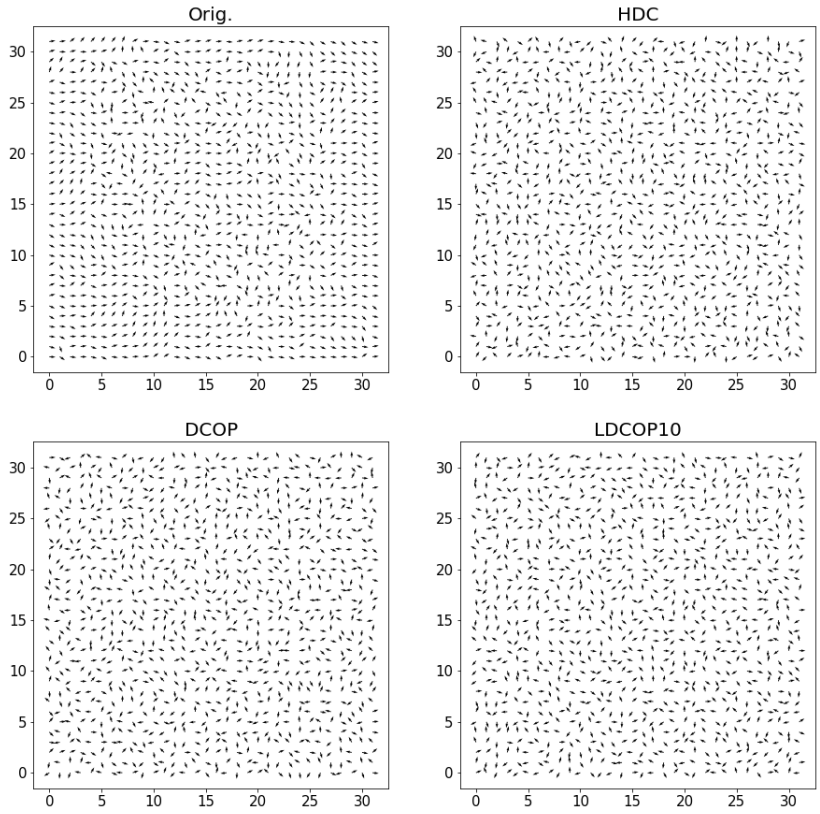


Figure 13 – Visualization of the approximate gradient (Equation 22), built by ZOO at its first algorithm iteration for a particular image and different defense strategies for the CNN. Each plot comprises a matrix of arrows representing an approximation of the loss function gradient concerning each input pixel. We fixed the arrow length, and its angle represents the gradient value for the pixel position. The defense is active except for the top leftmost plot.

3.4.3 Defense method visualization

Figure 14 presents a visualization of the ZOO adversarial attack against the fully connected neural network classifier, trained over the data set of concentric circles, with and without the application of the defense methods. We can see that the original input effortlessly turned into an adversarial example for the original neural network. However, the proposed disturbance approaches hampered the process of identifying the correct perturbation direction, making it harder to generate adversarial images and, in some cases, preventing the success of the attacker.

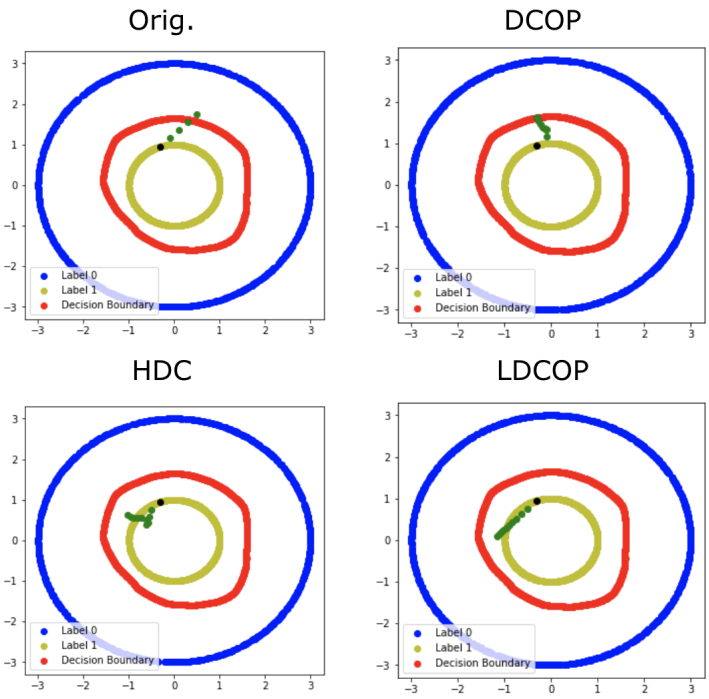


Figure 14 – Visualization of the ZOO attack iterations against the original neural network, with and without the application of the defensive methods. The blue and yellow circumferences consist of the training data set, the red curve comprises the decision boundary, the black dots represent the original sample where the attack begins and, the green dots form a trajectory during the ZOO attack procedure.

4 BENFORD'S LAW FOR ADVERSARIAL DETECTION ?

In this chapter, we show that adversarial images present a statistical property that differs when compared to natural images after applying proper mathematical transformations. More specifically, we analyze adversarial images concerning their First Digit Distribution.

4.1 OVERVIEW

Let $x \in R^{n \times m}$ be any two-dimensional image. We associate to each x a statistic s computed as follows: a) calculate the gradient magnitude of x, here denoted as the transformation $T(x)$; then b) get the frequency of the first digits of $T(x)$; c) compare, through the Kolmogorov-Smirnov statistic, the distribution got in (b) with the one given by Benford's law. The procedure encompassed by steps (a)-(c) is shown in Figure 15. We now detail each of these steps in the sections that follow.

4.1.1 Benford's Law

Benford's Law (BL) or the First Digit Law (BENFORD, 1938) states that, across different domains, the distribution of the leading digits of numerical data follows a similar pattern, namely, the one given by Equation 27:

$$P(d) = \log_{10}\left(1 + \frac{1}{d}\right) \tag{27}$$

where $d \in \{1, \ldots, 9\}$ represents the first digit value, and $P(d)$ the probability associated with each d.

Figure 16 portrays the First Digit distribution, as proposed in BL. Pixel-based

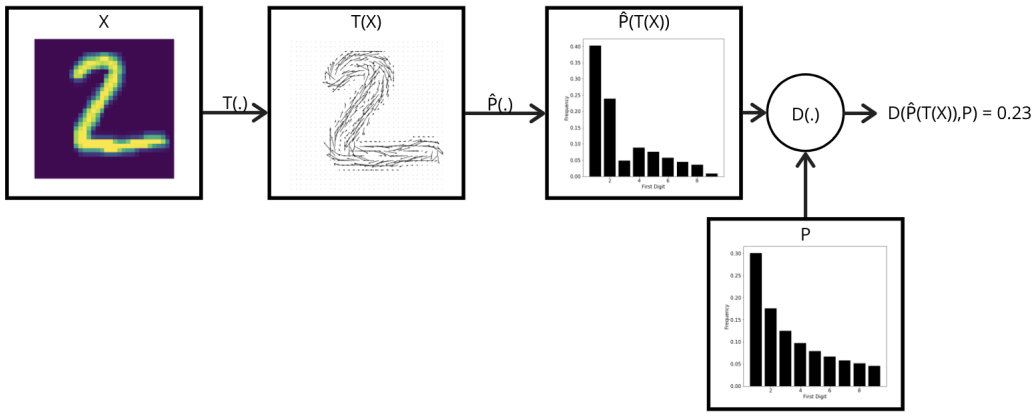


Figure 15 – Overview of the proposed approach, composed of three steps: a) transformation of the image x, represented by the mapping $T(.)$; b) a statistical analysis of $T(x)$, denoted by $\hat{P}$ and c) a comparison of $\hat{P}$ with a distribution of reference P.

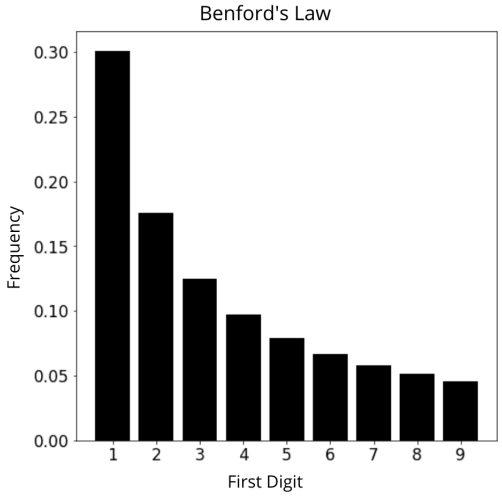


Figure 16 – First digit distribution as in Benford's Law

images rarely follow BL (JOLION, 2001). However, for certain transformations, the transformed images do. Two examples of such transformations are: the gradient magnitude of the input image (JOLION, 2001) and the Discrete Cosine Transformation (DCT) (PÉREZ-GONZÁLEZ; HEILEMAN; ABDALLAH, 2007). We employed the former as presented in Chapter 4 to map images such that the natural ones behave as in BL.

4.1.2 Image Transformation: Gradient Magnitude

Given an input image $x \in \mathbb{R}^{n \times m}$, we compute its gradient $G(x)$ according to Equation 28, below:

$$G(x)_{i,j} = \sqrt{G_{e_1}(x)_{i,j}^2 + G_{e_2}(x)_{i,j}^2}, \tag{28}$$

where $G(x)$ is the gradient magnitude; i and j are indices indicating each pixel value; $G_{e_1}(x)$ and $G_{e_2}(x)$ are the horizontal and vertical components of the gradient approximation, given by:

$$G_{e_1}(x) = x * K_{e_1}, G_{e_2}(x) = x * K_{e_2} \tag{29}$$

which are computed using the following Sobel filters K_{e_1} and K_{e_2} for the convolution operation with the input image:

$$K_{e_1} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, K_{e_2} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

The discrete convolution operation, employed in Equation 29 for the approximation of the gradient in both directions, is given by:

$$(x * K)_{i,j} = \sum_{o=1}^{O} \sum_{p=1}^{P} x_{i-o,j-p} K_{o,p} \tag{30}$$

where K represents, generically, both of the Sobel filters; O and P are the number of rows and columns, respectively, relative to K.

4.1.3 Computing the First Digit Distribution

Given the transformed image $T(x)$, we aim to calculate its associated first digit distribution. Firstly, we get the leading digit of each of its pixel values. The first digits of $T(x)$ are denoted as $F(T(x))$ (e.g., F will output 1 given the pixel value 176, and 5 given 54). Then the frequency for each digit $d \in \{1, \ldots, 9\}$ is computed, forming a distribution $\hat{P}(d)$, satisfying $\sum_{d=1}^{9} \hat{P}(d) = 1$. Note that $T(x)$ corresponds to $G(x)$ in this work.

4.1.4 Comparing two distributions: the Kolmogorov-Smirnov test

Given two distributions, one may use the Kolmogorov-Smirnov (KS) test to compute how much these distributions diverge. This test evaluates the distance between two empirical distributions or between the theoretical and an empirical distribution.

To assess the difference between the empirical distribution $\hat{P}$ and the theoretical distribution P, given by BL, we employed this test, by calculating the KS statistic between these distributions, denoted as $D^{KS}(\hat{P}, P)$, using the formula below:

$$D^{KS}(p, q) = \sup |Acc(p) - Acc(q)|, \tag{31}$$

where Acc returns the accumulated distribution of its given input distribution; and p and q correspond to the given distributions.

4.2 EXPERIMENTAL COMPONENTS

In this section, we present the experimental setup to evaluate the deviation of adversarial images regarding Benford's Law. We briefly present the selected image data sets and the employed CNN architecture and training parameters (when applied). The adversarial attacks used in our experiments are also listed.

4.2.1 Data sets

To evaluate this approach, we employed three different image data sets, listed as follows:

- MNIST;
- CIFAR10;
- ImageNet.

For further information, we described in detail these three data sets in Section 2.4.

4.2.2 CNNs under attack

We employed a different CNN model architecture for each of the data sets. We trained the CNN selected to classify images from the MNIST (Table 4) and CIFAR10 (VGG16 with the necessary adjustments due to some differences in the input size (SIMONYAN; ZISSERMAN, 2014)) data sets from scratch and employed a benchmark pre-trained network as the classifier for the images from ImageNet (VGG19 (SIMONYAN; ZISSERMAN, 2014)).

For the MNIST data set, the ADAM (KINGMA; BA, 2014) optimization algorithm was employed in the training procedure, with a learning rate of $1\mathrm{e}{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1\mathrm{e}{-08}$. We divided the data set into three different image sets: the training set comprising $50,000$ images, whereas the validation and test sets, $10,000$ samples each. The classifier was trained for about 50 epochs, reaching a training accuracy of about 99.29% and for the test set 97.03%.

Table 4 – Neural Network Architecture - MNIST

Layer	Type	Dimensions
0	Input	(32x32x3)
1	Conv(3x3)-64	(32x32x64)
2	Conv(3x3)-64	(32x32x64)
3	Batch Normalization	(32x32x64)
4	Max Pooling(2x2)	(16x16x64)
5	Conv(3x3)-128	(16x16x128)
6	Conv(3x3)-128	(16x16x128)
7	Batch Normalization	(16x16x128)
8	Max Pooling(2x2)	(8x8x128)
9	Conv(3x3)-256	(8x8x256)
10	Conv(3x3)-256	(8x8x256)
11	Batch Normalization	(8x8x256)
12	Max Pooling(2x2)	(4x4x256)
13	Fully Connected	(2048)
14	Batch Normalization	(2048)
15	Fully Connected	(2048)
16	Fully Connected	(10)

For training the classifier on the CIFAR10 data set, the optimization algorithm employed was the Gradient Descent with momentum, where the learning rate and the momentum values were $1\mathrm{e}{-3}$ and 0.9, respectively. We divided the data set into three different image sets: the training set comprising $40,000$ images, whereas the validation and test sets, $10,000$ samples each. The training procedure took about 50 epochs, reaching a training accuracy of about 99.74% and for the test set 80.11%.

4.2.3 Adversarial Attacks

We employed two different adversarial attack algorithms, the FGSM (GOODFELLOW; SHLENS; SZEGEDY, 2014) and the PGD (MADRY *et al.*, 2017). For the PGD attack, we applied two different settings: the $||.||_2 - norm$ and the $||.||_\infty - norm$. Both of the attacks are white-box. We presented a description of both adversarial attacks in Sections 2.4 and 2.4.

4.2.4 Experimental Description

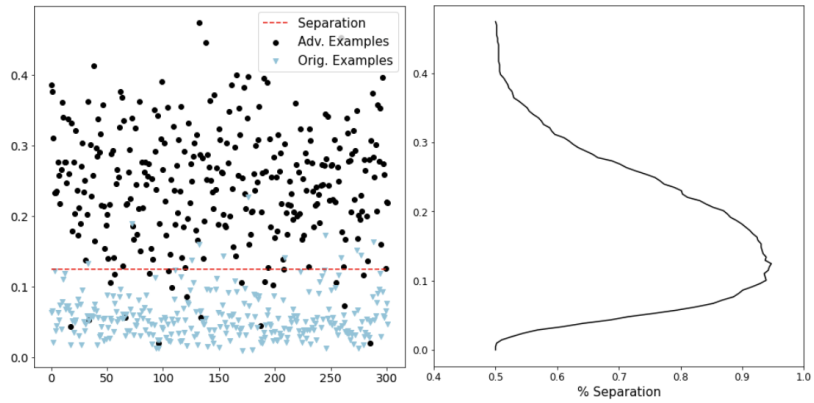
To analyze the deviation between the first digit distributions of the original and the adversarial images, we employed the following steps:

1. Select 1000 random images from the test set for each of the tested data sets (MNIST, CIFAR10, and ImageNet);
2. Attack each of these selected images with the employed adversarial attack algorithms (FGSM and PGD);
3. Compute transformation of the inputs by evaluating the gradient magnitude of the original, and the corresponding generated adversarial examples, according to Equation 28;
4. Compute the first digit distribution, $\hat{P}$, for the transformed input;
5. Evaluate the KS statistic for the first digit distribution of each original and adversarial example, compared to the original first digit distribution proposed by BL (Equation 31);
6. Evaluate the Kullback-Leibler (KL) divergence between the first digit distribution of both clean and adversarial examples, compared to the theoretical distribution presented in the FDL (Equation 32). As the KL divergence also evaluates the distance between distributions, this test enables its comparison regarding the Kolmogorov-Smirnov test.

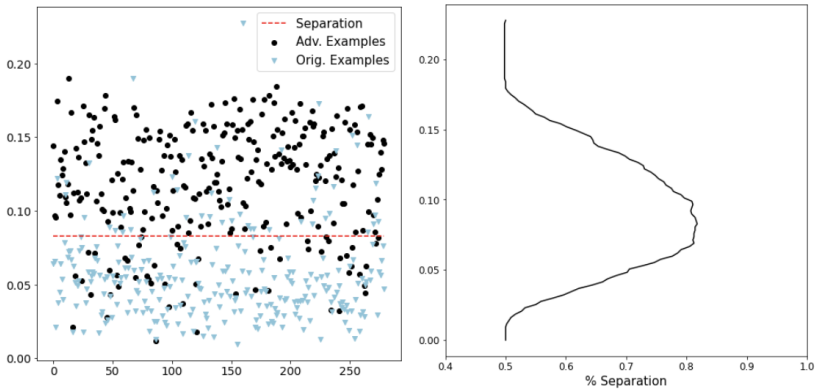
$$D^{KL}(p,q) = \sum_{d=1}^{9} p(d)\log(p(d) \div q(d)) \tag{32}$$

4.3 RESULTS

In this section, we show in detail the results reached by following the experimental procedure presented previously. We evaluate empirically the deviation of adversarial examples compared to clean inputs, compared these results while applying different



(a) Represents the adversarial examples generated using the $||.||_\infty - norm$ PGD attack



(b) Results associated with the examples crafted with the $||.||_2 - norm$ PGD attack

Figure 17 – Scatter plot of the KS statistic for adversarial and clean examples. The first column comprises the dispersion of the KS statistic for both adversarial and original examples. The second column contains the behavior of the separation percentage concerning the KS test value.

norms during the attack, analyzed the relation between the first digit distribution deviation and the perturbation magnitude, investigated the behavior of the computed deviation during the adversarial attack for PGD attack and compared the Kolmogorov-Smirnov test with the Kullback-Leibler divergence.

4.3.1 First digit distribution deviation

The comparison between the KS test for the adversarial examples and the original examples generated by the PGD using both the $||.||_\infty - norm$ and $||.||_2 - norm$ approaches, presented in Figure 17, indicates that most of the generated adversarial examples had a higher KS statistic value compared to the theoretical FDL.

We also have the relation of the separation percentage and the KS statistic for both cases, where it was possible to reach 94.7% for the adversarial examples generated by the $||.||_\infty - norm$ PGD attack, and 81.8% for the malicious inputs generated by the $||.||_2 - norm$ PGD attack. These results corroborate the fact that the first digit distribution of a transformed input image deviates from the adversarial examples associated with it.

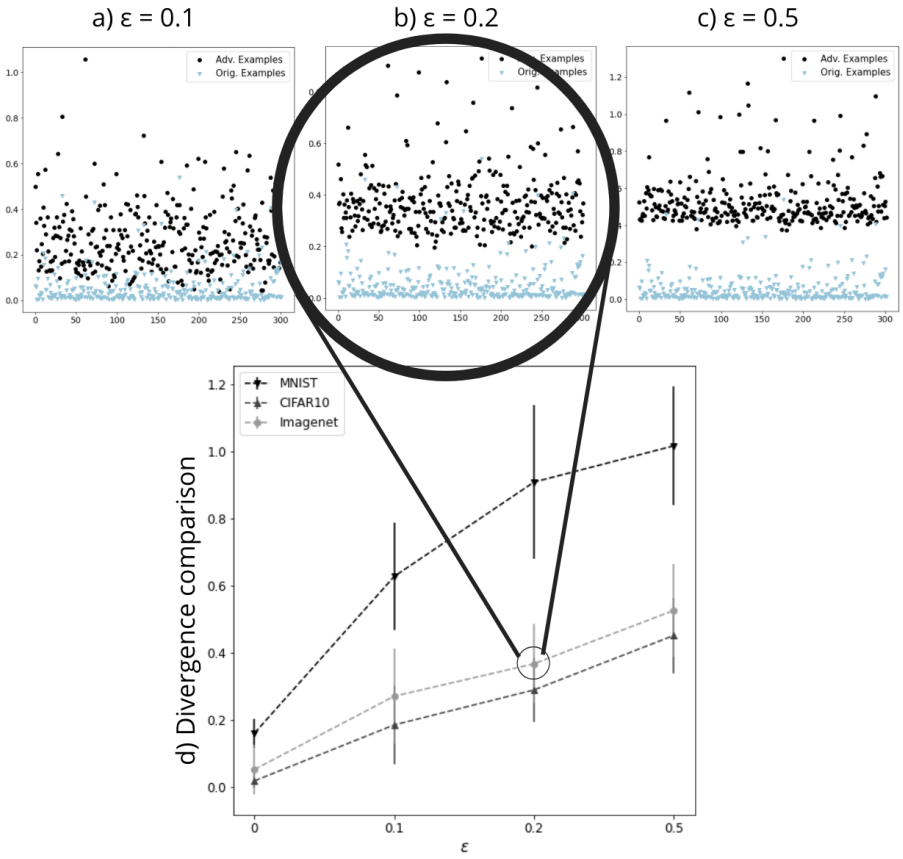


Figure 18 – Images a), b), and c) present the Kullback-Leibler divergence between the first digit distribution of the gradient magnitude for the adversarial and the original examples, concerning the original distribution proposed by Benford's Law, where we sampled the attacked images from the Imagenet data set. The adversarial examples were designed by the $||.||_\infty$-norm FGSM attack with ϵ equal to 0.1, 0.2 and 0.5, respectively. Image d) contains the mean and standard deviation for the KL-divergence while varying the ϵ for three different data sets.

4.3.2 Deviation regarding different attack norm

Figure 17 presents a comparison between the deviation calculated for adversarial examples generated by an $||.||_\infty - norm$ and a $||.||_2 - norm$ attacker. By analyzing the separation percentage for different values of the KS test, it is possible to visualize that the deviation of the $||.||_\infty - norm$ attacker is larger compared to the $||.||_2 - norm$, resulting in a higher reachable separation percentage. These results remained invariant for different image data sets.

4.3.3 Deviation regarding the perturbation magnitude

We have that the proposed approach makes it possible to separate adversarial examples from clean examples for different values of ϵ, as presented in Figure 18. Moreover, by comparing the divergence for higher values of ϵ, we notice that the

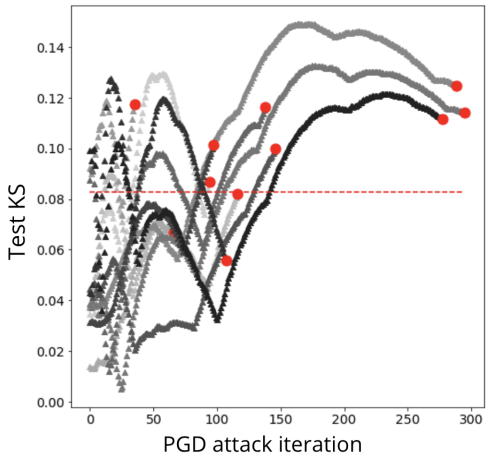


Figure 19 – Behavior of the KS test along the attack iteration of the $2-norm$ implementation of the PGD attack for 11 examples from the Imagenet data set.

higher the perturbation imposed to the original example, the more explicitly separated these sets become. This behavior is invariant for all the data sets employed during the experiments.

Despite the non-intuitive relation between the increment on the magnitude of the imposed perturbation and the deviation of the first digit distribution, the higher the perturbation designed, or artificiality crafted by the $||.||_\infty - norm$ attack, the easier it is to identify adversarial examples by comparing the first digit distribution with the distribution proposed by Benford's Law, as presented in this research.

4.3.4 Deviation regarding attack iteration

We have that the KS statistic overcame the separation limit computed according to the Figure 17 before turning into an adversarial example, for some of the examples presented in Figure 19. This limit consists of the separation value with a higher success rate for adversarial and clean inputs.

This result shows that, by analyzing the KS test value, it may be possible to identify that a neural network is under attack, i.e., that an attacker is using output information to design a perturbation that will change the classification associated with the original input image.

4.3.5 KS test compared to the KL divergence

We employed the PGD attack, under $||.||_2 - norm$ and $||.||_\infty - norm$, for comparing the maximum separation percentage while using the KS test and the KL divergence as it is considered the universal first-order attack (MADRY *et al.*, 2017). The results presented in Table 5 indicate a significant increment in the separation percentage when applying the KS test under both PGD settings.

Table 5 – Maximum separation percentage comparison between the Kullback-Leibler divergence and the Kolmogorov-Smirnov statistics.

	KL-divergence	KS test
PGD $\infty - norm$	90.23%	94.70%
PGD $2 - norm$	66.96%	81.79%

This analysis suggests that the KS test is more sensitive to the deviations on the first digit distribution because it enables reaching a higher separation percentage using the same amount of information given for the KL-divergence.

5 CONCLUSION

In this work, we have proposed two different approaches to deal with adversarial examples focused on the defensive side of the digital arms race between adversarial attacks and defenses. The first approach consists of a novel defense strategy against black-box adversarial attacks. It comprises adding a controlled random disturbance to the softmax output layer of a neural network classifier. The second consists of a new approach by which one can use to create a detector of adversarial images. Our approach relies on computing the first digit distribution of the pixel values for the classifier's input samples, assuming that adversarial examples do not follow Benford's law (BL) as natural images do, when transformed. Both methods act solely in the prediction phase. They do not change the training data set or the classifiers parameters and training process.

For the defense method against black-box adversarial attacks, we have designed three different approaches to compute such disturbance. The difference among them lies in the set of requirements the disturbance vector must satisfy.

To evaluate our defensive approach, we have trained a CNN on the MNIST data set and attacked it with the ZOO algorithm. Our findings show that the proposed defense methods make the trained CNN less vulnerable to the images crafted by ZOO. Our defense method reduces the attack success rate of ZOO while keeping the utility of the network's predictions (i.e., the general form of the probability distribution over classes). For instance, the LDCOP10 method (which keeps the class ordering of the softmax output and limits the noise magnitude) still attains a significant reduction in the attack success rate generated by ZOO from 99.8% to 65.3%.

Our defense methods force the attacker to increase the magnitude of the perturbation necessary to cause a misclassification. Therefore, if the network misclassifies an image generated by the attacker, it is probable that this image either: is not adversarial due to its high perturbation or is detectable by other defense methods (LU; ISSARANON; FORSYTH, 2017; METZEN *et al.*, 2017; FEINMAN *et al.*, 2017; GROSSE *et al.*, 2017; SONG *et al.*, 2017).

The combination of the proposed method with existing algorithms, left as future work, would result in a strong defense since they are orthogonal: while most methods depend on augmenting the training data set or retraining the network, ours act exclusively on the network prediction and does not explicitly depend on a particular data set. Moreover, because of its simplicity and ease of application, we advocate using the proposed strategy as a standard method to evaluate the robustness of new black-box attack methods.

The second proposed approach comprises computing the Kolmogorov-Smirnov statistic between the first digit distribution of CNN's input image and the fixed distribution from BL (after applying a suitable transformation to the given image). Specifically,

we have shown that the leading digit distribution of adversarial images generated by FGSM and PGD attack methods differ significantly from the corresponding distribution observed in unaltered images: the former deviates significantly more from BL when compared to the latter. This deviation tends to become higher as the perturbation's magnitude increases (for the FGSM attack).

Besides, one can use our approach to anticipate a potential undergoing attack since we have observed that, in many cases, the deviation given by the KS statistic reaches the separation threshold before the image becomes adversarial.

5.1 FUTURE WORKS

Additionally, we plan to investigate the performance of our defenses against black-box adversarial attacks in different settings, such as: a) against other black-box attack algorithms that depend on the output of the softmax layer; b) when using a classifier trained on different data sets with a higher number of features and output labels (e.g., CIFAR-10, CIFAR-100); c) when different probability distributions (such as the uniform distribution) are employed to compute the required controlled disturbance; d) when combining the proposed defense method with existing adversarial defensive approaches (those that increase the classifier's robustness and those that detect malicious examples).

Regarding the statistical study of adversarial images, we left as future works: a) devising a sophisticated adversarial image detector, based on the output of the KS statistic test, which can serve as a low-dimensional input feature instead of the whole high-dimensional image; b) divide an image such that multiple KS statistics tests are obtained from a given input image, providing a more refined, informative view of the deviation caused by the attack perturbation; c) consider different adversarial attack methods, mainly black-box algorithms or those that perturb only a few pixels; d) investigate the iterative application of the gradient transformation over a given image (e.g., two times) due to preliminary tests that showed an improvement in the results while employing the KL divergence instead of the KS test.

5.2 PUBLICATIONS

The studies in this work resulted in the following publications:

- Zago J. G., Antonelo E. A., Baldissera F. L. and Saad R. T.. It is double pleasure to deceive the deceiver: disturbing classifiers against adversarial attacks. 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), p. 160-165. DOI: 10.1109/SMC42975.2020.9282889;

- Zago J. G., Antonelo E. A., Baldissera F. L. and Saad R. T.. Benford's law: what does it say on adversarial images? arXiv preprint: 2102.04615 (Submitted to Neural Processing Letters journal).

REFERENCES

ATHALYE, Anish *et al.* Synthesizing Robust Adversarial Examples, July 2017.

BENFORD, Frank. The law of anomalous numbers. **Proceedings of the American philosophical society**, JSTOR, p. 551–572, 1938.

BROWN, Tom *et al.* Adversarial Patch, Dec. 2017.

CARLINI, Nicholas; WAGNER, David. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods, May 2017.

CARLINI, Nicholas; WAGNER, David. Towards Evaluating the Robustness of Neural Networks. *In:* p. 39–57. DOI: 10.1109/SP.2017.49.

CHEN, Pin-Yu *et al.* ZOO: Zeroth Order Optimization Based Black-box Attacks to Deep Neural Networks without Training Substitute Models. *In:* p. 15–26. DOI: 10.1145/3128572.3140448.

DECKERT, Joseph; MYAGKOV, Mikhail; ORDESHOOK, Peter C. Benford's Law and the detection of election fraud. **Political Analysis**, Cambridge University Press, v. 19, n. 3, p. 245–268, 2011.

DENG, J. *et al.* ImageNet: A Large-Scale Hierarchical Image Database. *In:* CVPR09. [*S.l.*: *s.n.*], 2009.

FEINMAN, Reuben *et al.* Detecting adversarial samples from artifacts. **arXiv preprint arXiv:1703.00410**, 2017.

GOODFELLOW, Ian J.; SHLENS, Jonathon; SZEGEDY, Christian. Explaining and Harnessing Adversarial Examples. **CoRR**, abs/1412.6572, 2014.

GROSSE, Kathrin *et al.* On the (statistical) detection of adversarial examples. **arXiv preprint arXiv:1702.06280**, 2017.

GU, Shixiang; RIGAZIO, Luca. Towards deep neural network architectures robust to adversarial examples. **arXiv preprint arXiv:1412.5068**, 2014.

HINTON, Geoffrey; DENG, li, *et al.* Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. **Signal Processing Magazine, IEEE**, v. 29, p. 82–97, Nov. 2012. DOI: 10.1109/MSP.2012.2205597.

HINTON, Geoffrey; VINYALS, Oriol; DEAN, Jeff. Distilling the knowledge in a neural network. **arXiv preprint arXiv:1503.02531**, 2015.

HOSSEINI, Hossein; POOVENDRAN, Radha. Semantic Adversarial Examples, Mar. 2018.

HUANG, Xiaowei *et al.* Safety Verification of Deep Neural Networks. English. Ed. by Rupak Majumdar and Viktor Kunčak. Springer, Part 1, p. 3–29, July 2017. DOI: 10.1007/978-3-319-63387-9_1.

JOLION, Jean-Michel. Images and Benford's law. **Journal of Mathematical Imaging and Vision**, Springer, v. 14, n. 1, p. 73–81, 2001.

KATZ, Guy *et al.* Reluplex: An efficient SMT solver for verifying deep neural networks. *In:* SPRINGER. INTERNATIONAL Conference on Computer Aided Verification. [*S.l.*: *s.n.*], 2017. P. 97–117.

KHOURY, Marc; HADFIELD-MENELL, Dylan. **On the Geometry of Adversarial Examples**. [*S.l.*: *s.n.*], 2019. Disponível em: https://openreview.net/forum?id=H1lug3R5FX.

KINGMA, Diederik P; BA, Jimmy. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014.

KRIZHEVSKY, Alex; HINTON, Geoffrey. Learning Multiple Layers of Features from Tiny Images. **University of Toronto**, May 2012.

KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey. ImageNet Classification with Deep Convolutional Neural Networks. **Neural Information Processing Systems**, v. 25, Jan. 2012.

KURAKIN, Alexey; GOODFELLOW, Ian; BENGIO, Samy. Adversarial examples in the physical world, July 2016.

KURAKIN, Alexey; GOODFELLOW, Ian; BENGIO, Samy. Adversarial Machine Learning at Scale, Nov. 2016.

LECUN, Yann; BOTTOU, Léon, *et al.* Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, Ieee, v. 86, n. 11, p. 2278–2324, 1998.

LECUN, Yann; CORTES, Corinna. MNIST handwritten digit database, 2010. Disponível em: http://yann.lecun.com/exdb/mnist/.

LIU, Changliu *et al.* Algorithms for verifying deep neural networks. **arXiv preprint arXiv:1903.06758**, 2019.

LU, Jiajun; ISSARANON, Theerasit; FORSYTH, David. SafetyNet: Detecting and Rejecting Adversarial Examples Robustly. *In:* p. 446–454. DOI: 10.1109/ICCV.2017.56.

LU, Jiajun; SIBAI, Hussein, *et al.* NO Need to Worry about Adversarial Examples in Object Detection in Autonomous Vehicles, July 2017.

MADRY, Aleksander *et al.* Towards deep learning models resistant to adversarial attacks. **arXiv preprint arXiv:1706.06083**, 2017.

MENG, Dongyu; CHEN, Hao. Magnet: a two-pronged defense against adversarial examples. *In:* PROCEEDINGS of the 2017 ACM SIGSAC Conference on Computer and Communications Security. [*S.l.*: *s.n.*], 2017. P. 135–147.

METZEN, Jan Hendrik *et al.* On detecting adversarial perturbations. **arXiv preprint arXiv:1702.04267**, 2017.

MILANI, Simone *et al.* Phylogenetic analysis of near-duplicate images using processing age metrics. *In:* IEEE. 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). [*S.l.*: *s.n.*], 2016. P. 2054–2058.

MOOSAVI-DEZFOOLI, Seyed-Mohsen; FAWZI, Alhussein; FAWZI, Omar, *et al.* Universal Adversarial Perturbations. *In:* p. 86–94. DOI: 10.1109/CVPR.2017.17.

MOOSAVI-DEZFOOLI, Seyed-Mohsen; FAWZI, Alhussein; FROSSARD, Pascal. DeepFool: a simple and accurate method to fool deep neural networks. **CVPR**, Nov. 2016.

NA, Taesik; KO, Jong Hwan; MUKHOPADHYAY, Saibal. Cascade Adversarial Machine Learning Regularized with a Unified Embedding. *In:* INTERNATIONAL Conference on Learning Representations. [*S.l.*: *s.n.*], 2018. Disponível em: https://openreview.net/forum?id=HyRVBzap-.

NARODYTSKA, Nina; KASIVISWANATHAN, Shiva Prasad. Simple black-box adversarial perturbations for deep networks. **arXiv preprint arXiv:1612.06299**, 2016.

PAPERNOT, Nicolas; MCDANIEL, Patrick; GOODFELLOW, Ian, *et al.* Practical Black-Box Attacks against Machine Learning. *In:* p. 506–519. DOI: 10.1145/3052973.3053009.

PAPERNOT, Nicolas; MCDANIEL, Patrick; WU, Xi, *et al.* Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. *In:* p. 582–597. DOI: 10.1109/SP.2016.41.

PAPERNOT, Nicolas; MCDANIEL, Patrick D.; JHA, Somesh, *et al.* The Limitations of Deep Learning in Adversarial Settings. **2016 IEEE European Symposium on Security and Privacy (EuroS&P)**, p. 372–387, 2015.

PÉREZ-GONZÁLEZ, Fernando; HEILEMAN, Greg L; ABDALLAH, Chaouki T. Benford's law in image processing. *In:* IEEE. 2007 IEEE International Conference on Image Processing. [*S.l.*: *s.n.*], 2007. P. i–405.

PEVNY, Tomas; FRIDRICH, Jessica. Detection of double-compression in JPEG images for applications in steganography. **IEEE Transactions on information forensics and security**, IEEE, v. 3, n. 2, p. 247–258, 2008.

RUMELHART, David E.; HINTON, Geoffrey E.; WILLIAMS, Ronald J. Learning Representations by Back-propagating Errors. **Nature**, v. 323, n. 6088, p. 533–536, 1986. DOI: 10.1038/323533a0. Disponível em: http://www.nature.com/articles/323533a0.

SABOUR, Sara *et al.* Adversarial manipulation of deep representations. **arXiv preprint arXiv:1511.05122**, 2015.

SIMONYAN, Karen; ZISSERMAN, Andrew. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014.

SONG, Yang *et al.* Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. **arXiv preprint arXiv:1710.10766**, 2017.

SU, Jiawei; VARGAS, Danilo; SAKURAI, Kouichi. One Pixel Attack for Fooling Deep Neural Networks. **IEEE Transactions on Evolutionary Computation**, PP, Oct. 2017. DOI: 10.1109/TEVC.2019.2890858.

SZEGEDY, Christian; IOFFE, Sergey, *et al.* Inception-v4, inception-resnet and the impact of residual connections on learning. *In:* 1. PROCEEDINGS of the AAAI Conference on Artificial Intelligence. [*S.l.*: *s.n.*], 2017.

SZEGEDY, Christian; LIU, Wei, *et al.* Going deeper with convolutions. *In:* PROCEEDINGS of the IEEE conference on computer vision and pattern recognition. [*S.l.*: *s.n.*], 2015. P. 1–9.

SZEGEDY, Christian; VANHOUCKE, Vincent, *et al.* Rethinking the Inception Architecture for Computer Vision. *In:* DOI: 10.1109/CVPR.2016.308.

SZEGEDY, Christian; ZAREMBA, Wojciech, *et al.* Intriguing properties of neural networks. **CoRR**, abs/1312.6199, 2013.

TJENG, Vincent; XIAO, Kai; TEDRAKE, Russ. Evaluating robustness of neural networks with mixed integer programming. **arXiv preprint arXiv:1711.07356**, 2017.

TÖDTER, Karl-Heinz. Benford's Law as an Indicator of Fraud in Economics. **German Economic Review**, Wiley Online Library, v. 10, n. 3, p. 339–351, 2009.

TRAMÈR, Florian *et al.* Ensemble Adversarial Training: Attacks and Defenses. *In:* INTERNATIONAL Conference on Learning Representations. [*S.l.*: *s.n.*], 2018. Disponível em: `https://openreview.net/forum?id=rkZvSe-RZ`.

XIANG, Weiming; TRAN, Hoang-Dung; JOHNSON, Taylor T. Reachable set computation and safety verification for neural networks with relu activations. **arXiv preprint arXiv:1712.08163**, 2017.

ZHAO, Zhengli; DUA, Dheeru; SINGH, Sameer. Generating Natural Adversarial Examples. **ICLR**, Feb. 2018.