

MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE MESTRADO EM SISTEMAS E COMPUTAÇÃO

GABRIEL RESENDE MACHADO

MULTIMAGNET: UMA ABORDAGEM BASEADA NA
FORMAÇÃO NÃO DETERMINÍSTICA DE COMITÊS PARA
DETECÇÃO DE IMAGENS CONTRADITÓRIAS

Rio de Janeiro
2019

INSTITUTO MILITAR DE ENGENHARIA

GABRIEL RESENDE MACHADO

***MULTIMAGNET: UMA ABORDAGEM BASEADA NA
FORMAÇÃO NÃO DETERMINÍSTICA DE COMITÊS PARA
DETECÇÃO DE IMAGENS CONTRADITÓRIAS***

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Mestre em Ciências em Sistemas e Computação.

Orientador: Prof. Ronaldo Ribeiro Goldschmidt - D.Sc.

Co-Orientador: Prof. Eugênio Silva - D.Sc.

Rio de Janeiro
2019

CIP - Catalogação na Publicação

Resende Machado, Gabriel
MultiMagNet: Uma Abordagem Baseada na Formação
Não Determinística de Comitês para Detecção de
Imagens Contraditórias / Gabriel Resende Machado.
- Rio de Janeiro, 2019.
86 f.

Orientador: Ronaldo Ribeiro Goldschmidt.
Coorientador: Eugênio Silva.
Dissertação (mestrado) - Instituto Militar de
Engenharia, Sistemas e Computação, 2019.

1. Aprendizado Profundo. 2. Imagens
Contraditórias. 3. Não Determinismo. I. Ribeiro
Goldschmidt, Ronaldo , orient. II. Silva,
Eugênio, coorient. III. Título.

INSTITUTO MILITAR DE ENGENHARIA

GABRIEL RESENDE MACHADO

**MULTIMAGNET: UMA ABORDAGEM BASEADA NA
FORMAÇÃO NÃO DETERMINÍSTICA DE COMITÊS PARA
DETECÇÃO DE IMAGENS CONTRADITÓRIAS**

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Mestre em Ciências em Sistemas e Computação.

Orientador: Prof. Ronaldo Ribeiro Goldschmidt - D.Sc.

Co-Orientador: Prof. Eugênio Silva - D.Sc.

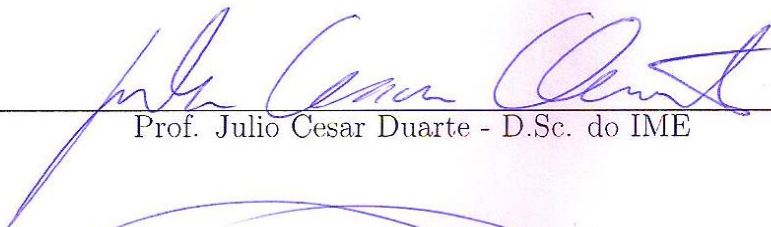
Aprovada em 26 de Abril de 2019 pela seguinte Banca Examinadora:



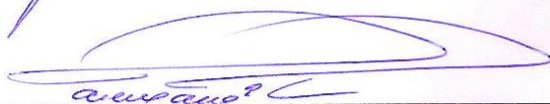
Prof. Ronaldo Ribeiro Goldschmidt - D.Sc. do IME - Presidente



Prof. Eugênio Silva - D.Sc. da UEZO



Prof. Julio Cesar Duarte - D.Sc. do IME



Prof. Marley Maria Bernardes Rebuzzi Vellasco - Ph.D. da PUC-Rio

Rio de Janeiro
2019

Dedico este trabalho à Jesus Cristo, aos meus pais, Valdir Corrêa Machado e Ana Maria Machado, e à minha esposa, Sabrina Gomes.

AGRADECIMENTOS

Agradeço, em especial, ao Deus revelado em Jesus Cristo, porque dEle, por Ele e para Ele são todas as coisas. Tudo o que existe vem dEle e todas as coisas subsistem por Ele. Louvado seja Seu Nome para sempre.

Aos meus pais, Valdir Corrêa Machado e Ana Maria Machado, pelo amor, carinho e apoio que, desde a minha infância, vêm formando meu caráter e me conduzindo a patamares mais elevados.

À minha esposa, Sabrina Gomes, pelo constante apoio, compreensão e paciência nos momentos mais difíceis.

Aos meus professores e orientadores Ronaldo Ribeiro Goldschmidt e Eugênio Silva que, com enorme sabedoria, carinho, paciência e disponibilidade me instruíram ao longo deste curso. Deixo aqui manifesta minha eterna gratidão e parceria.

Ao Instituto Militar de Engenharia pela oportunidade concedida e à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pela bolsa de mestrado (código de financiamento 001).

“From life’s first cry to final breath, Jesus commands my destiny.”

TRECHO DO HINO *IN CHRIST ALONE*.

SUMÁRIO

LISTA DE ILUSTRAÇÕES	10
LISTA DE TABELAS	12
LISTA DE SIGLAS	13
LISTA DE ABREVIATURAS	14
1 INTRODUÇÃO	17
1.1 Contextualização e Motivação	17
1.2 Problema	18
1.3 Hipótese e Objetivo	19
1.4 Contribuições Esperadas	20
1.5 Organização do Trabalho	20
2 CONCEITOS BÁSICOS	21
2.1 Aprendizado de Máquina	21
2.2 Aprendizado Profundo	22
2.2.1 Redes Neurais Convolucionais	22
2.2.1.1 Camada de convolução	23
2.2.1.2 Camada de <i>pooling</i>	24
2.2.1.3 Camada densamente conectada	25
2.2.2 <i>Autoencoders</i>	26
2.3 Imagens Contraditórias	27
2.3.1 Taxonomia das Imagens Contraditórias	29
2.3.1.1 Escopo da perturbação	29
2.3.1.2 Visibilidade da perturbação	30
2.3.1.3 Medida da perturbação	30
2.3.2 Taxonomia dos Ataques Contraditórios	31
2.3.2.1 Influência do atacante	31
2.3.2.2 Conhecimento do atacante	32
2.3.2.3 Especificidade do ataque	34
2.3.2.4 Cálculo do ataque	34
2.3.2.5 Abordagem do ataque	34
2.3.3 Algoritmos de ataques contraditórios	35

2.3.3.1	<i>Fast Gradient Sign Method</i>	36
2.3.3.2	<i>Basic Iterative Method</i>	36
2.3.3.3	<i>DeepFool</i>	37
2.3.3.4	<i>Carlini & Wagner Attack</i>	37
2.4	Divergência de <i>Jensen-Shannon</i>	39
3	TRABALHOS RELACIONADOS	40
3.1	Considerações Iniciais	40
3.2	Métodos Baseados em <i>Gradient Masking</i>	40
3.3	Métodos Baseados em Comitês de Classificadores	43
3.4	Modelos Auxiliares Detectores	43
3.5	Métodos Baseados em Abordagens Estatísticas	45
3.6	Métodos Baseados em Pré-Processamento	46
3.7	Métodos Baseados em Proximidade	48
4	O MULTIMAGNET	53
4.1	Visão Geral	53
4.2	Etapa de Calibração	53
4.2.1	Calcular Limiares	54
4.2.2	Formar Comitê (Calibração)	56
4.2.3	Avaliar Desempenhos	56
4.2.4	Selecionar Melhor Desempenho	57
4.3	Etapa de Produção	58
4.3.1	Formar Comitê (Produção)	59
4.3.2	Reconstruir Imagem	59
4.3.3	Calcular Métrica	59
4.3.4	Apurar Votos	59
4.3.5	Reformar Imagem	59
5	EXPERIMENTOS E RESULTADOS	61
5.1	Visão Geral	61
5.2	<i>Datasets</i>	62
5.3	Particionamento dos dados	63
5.4	Protótipo	64
5.5	Resultados	65
5.5.1	Conjunto de Experimentos I	66

5.5.2	Conjunto de Experimentos II	68
6	CONSIDERAÇÕES FINAIS	70
7	REFERÊNCIAS BIBLIOGRÁFICAS	72
8	APÊNDICES	81
8.1	APÊNDICE 1: Arquiteturas e hiperparâmetros dos classificadores	82
8.2	APÊNDICE 2: Arquiteturas e hiperparâmetros dos <i>autoencoders</i>	84
8.3	APÊNDICE 3: Hiperparâmetros definidos para a realização dos ataques contraditórios	86

LISTA DE ILUSTRAÇÕES

FIG.1	Desempenho das arquiteturas de RNC vencedoras do desafio <i>ILS-VRC top-5 error rate</i> durante os anos.	18
FIG.2	Um ataque contraditório ao sistema responsável pelo reconhecimento de imagens de um veículo autônomo poderia acarretar em acidentes de trânsito.	18
FIG.3	O processo de aprendizagem de um algoritmo de AM em uma abordagem supervisionada.	21
FIG.4	Exemplo de uma RNC com várias camadas internas.	23
FIG.5	O processo de convolução feito em uma RNC.	24
FIG.6	Exemplo de uma janela de <i>max-pooling</i> com <i>stride</i> 2×2	25
FIG.7	Um exemplo de <i>autoencoder</i> . Ao receber uma imagem de entrada, o <i>autoencoder</i> a reconstrói a partir da representação comprimida aprendida pela etapa de codificação.	27
FIG.8	Perturbações imperceptíveis, inseridas maliciosamente em uma imagem, podem induzir um modelo de classificação ao erro.	28
FIG.9	O objetivo de um ataque contraditório é gerar uma perturbação δx e inserí-la em uma imagem x a ponto de fazer com que x cruze a fronteira de decisão, e desta forma, produza um erro de classificação.	28
FIG.10	Taxonomia proposta para as imagens contraditórias.	29
FIG.11	Taxonomia proposta para os ataques contraditórios.	31
FIG.12	Ataques com influência causativa e evasiva em um modelo de classificação binário.	32
FIG.13	O <i>Adversarial Training</i> aumenta a robustez dos classificadores ao utilizar a técnica do <i>data augmentation</i>	41
FIG.14	Modelo esquemático da Destilação Defensiva.	42
FIG.15	O funcionamento do <i>Feature Squeezing</i>	48
FIG.16	Modelo esquemático do funcionamento de uma DkNN.	49
FIG.17	Ilustração de um Classificador baseado em Região.	49
FIG.18	O modelo esquemático do <i>MagNet</i>	51
FIG.19	Modelo esquemático da Etapa de Calibração.	54
FIG.20	Modelo esquemático da Etapa de Produção.	58

FIG.21	Exemplos de imagens do MNIST. A primeira linha contém imagens legítimas. As demais apresentam as versões perturbadas dessas imagens pelos ataques FGSM, BIM, <i>DeepFool</i> e CW, respectivamente.	62
FIG.22	Exemplos de imagens do CIFAR-10. A primeira linha contém imagens legítimas. As demais apresentam as versões perturbadas dessas imagens pelos ataques FGSM, BIM, <i>DeepFool</i> e CW, respectivamente.	63

LISTA DE TABELAS

TAB.1	Os principais algoritmos de ataques contraditórios.	38
TAB.2	Acurácias do classificador da aplicação nos conjuntos T_e e D_{atk} , sem nenhuma defesa presente.	66
TAB.3	O conjunto de hiperparâmetros c_b definido para o <i>MagNet</i> e <i>MultiMagNet</i> em cada algoritmo de ataque.	66
TAB.4	Média e desvio padrão dos resultados obtidos pelo <i>MagNet</i> e <i>MultiMagNet</i> em 40 experimentos no <i>dataset</i> MNIST.	67
TAB.5	Média e desvio padrão dos resultados obtidos pelo <i>MagNet</i> e <i>MultiMagNet</i> em 40 experimentos no <i>dataset</i> CIFAR-10.	67
TAB.6	Acurácias obtidas pelo classificador da aplicação F quando avaliado no conjunto D em cinco Cenários de ataque diferentes.	69
TAB.7	Arquitetura e hiperparâmetros definidos para o classificador da aplicação no <i>dataset</i> MNIST.	82
TAB.8	Arquitetura e hiperparâmetros definidos para o classificador da aplicação no <i>dataset</i> CIFAR-10.	83
TAB.9	Arquiteturas e hiperparâmetros adotados para cada <i>autoencoder</i> do repositório S nos experimentos com o <i>dataset</i> MNIST.	84
TAB.10	Arquiteturas e hiperparâmetros adotados para cada <i>autoencoder</i> do repositório S nos experimentos com o <i>dataset</i> CIFAR-10.	85
TAB.11	Hiperparâmetros definidos para cada algoritmo de ataque.	86

LISTA DE SIGLAS

AM	Aprendizado de Máquina
AP	Aprendizado Profundo
RNC	Rede Neural Convolutacional
JSD	Divergência de <i>Jensen-Shannon</i>
KL	Divergência de <i>Kullback-Leibler</i>
GAN	Rede Geradora Contraditória
DCN	Rede Contrativa Profunda
kNN	<i>k Nearest Neighbors</i>
RC	Classificadores Baseados em Região
MAD	Modelo Auxiliar Detector
MMD	<i>Maximum Mean Discrepancy</i>
KDE	<i>Kernel Density Estimation</i>
PCA	<i>Principal Component Analysis</i>
ACC	Acurácia
VPP	Valor Preditivo Positivo
VPN	Valor Preditivo Negativo
CE	Conjunto de Experimentos
MSE	Erro Médio Quadrático
ART	<i>Adversarial Robustness Toolbox</i>
FGSM	<i>Fast Gradient Sign Method</i>
BIM	<i>Basic Iterative Method</i>
CW	<i>Carlini & Wagner Attack</i>
DDoS	<i>Distributed Denial of Service</i>
JSMA	<i>Jacobian-based Saliency Map Attack</i>
LaVAN	<i>Localized and Visible Adversarial Noise</i>
L-BFGS	<i>Limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm</i>
ILLC	<i>Iterative Least-Likely Class attack</i>
PGD	<i>Projected Gradient Descent</i>
ZOO	<i>Zeroth-Order Optimization</i>
ATN	<i>Adversarial Transformation Networks</i>
GDE	Gradiente Descendente Estocástico
LSTM	<i>Long Short-Term Memory</i>

LISTA DE ABREVIATURAS E SÍMBOLOS

ABREVIATURAS

i.e. - *id est* (ou seja)

SÍMBOLOS

x - imagem legítima

x' - imagem contraditória

\hat{x} - imagem reconstruída

F - saídas *softmax* do modelo classificador

Z - saída *logits* do modelo classificador

RESUMO

Os algoritmos de Aprendizado Profundo, em especial as Redes Neurais Convolucionais, são o estado da arte na solução de diversas tarefas envolvendo classificação e reconhecimento de imagens. Entretanto, trabalhos recentes mostraram que esses algoritmos estão suscetíveis a ataques com imagens contraditórias. Imagens contraditórias são aquelas que possuem perturbações ínfimas, geradas maliciosamente por um algoritmo de ataque, com o objetivo de produzir erros de classificação. Essas imagens maliciosas ameaçam seriamente a adoção de modelos de aprendizado em tarefas críticas de segurança em Visão Computacional, uma vez que ataques conduzidos nesses ambientes de produção poderiam acarretar em acidentes e prejuízos de larga escala. Este cenário problemático incentivou a comunidade científica a propor inúmeros métodos para defesa de modelos de aprendizado contra imagens contraditórias. Contudo, os principais métodos de defesa existentes têm falhado principalmente por permitirem que o atacante compreenda seus *modi operandi*. Assim sendo, esta dissertação propõe o *MultiMagNet*, um método de defesa que, ao apresentar um comportamento não determinístico, dificulta evasões por ataques contraditórios. O não determinismo do método proposto decorre de uma escolha aleatória, realizada em tempo de execução, de um comitê de múltiplos componentes que analisam individualmente as imagens a fim de, em conjunto, identificar exemplares contraditórios. Os resultados de um estudo comparativo conduzido nos *datasets* MNIST e CIFAR-10 mostraram que o *MultiMagNet* foi capaz de superar o *MagNet*, um método de detecção de imagens contraditórias baseado em comportamento não determinístico, na defesa contra imagens contraditórias geradas pelos principais algoritmos de ataque.

ABSTRACT

Deep Learning algorithms, especially the Convolutional Neural Networks, are the state-of-the-art for solving several tasks involving image classification and recognition. However, recent works have shown that Convolutional Neural Networks are susceptible to attacks with adversarial images. Adversarial images have tiny perturbations, maliciously generated using an attack algorithm, in order to lead pretrained models to misclassification. The adversarial images seriously menace the use of deep learning models in security-critical tasks involving Computer Vision, since successful attacks conducted in these environments can result in large-scale losses and accidents. This alarming scenario though, has encouraged the scientific community to propose various methods to defend learning models against adversarial images. Nevertheless, the main existing defense methods have failed, especially because they facilitate the attacker to comprehend their *modi operandi*. Therefore, this dissertation proposes *MultiMagNet*, a defense method that, due to its non-deterministic behaviour, hinders evasions by adversarial attacks. The non-deterministic effect of the proposed defense method stems from an ensemble randomly formed at runtime, containing multiple defense components, which individually analyze the input images in order to identify adversarial samples. The results obtained from a comparative study conducted on the datasets MNIST and CIFAR-10 have shown that *MultiMagNet* was able to surpass *MagNet*, a detection method based on non-deterministic behaviour, when defending against adversarial images generated by the main attack algorithms.

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO E MOTIVAÇÃO

Com o crescimento exponencial na quantidade e complexidade dos dados produzidos anualmente e a necessidade de realizar a engenharia de atributos sem o auxílio de especialistas na área, foi preciso evoluir as tradicionais técnicas de Aprendizado de Máquina (AM) em algoritmos mais elaborados, que formam uma área recente de AM conhecida como Aprendizado Profundo (AP). A área de Aprendizado Profundo vem evoluindo e ganhando importância em uma escala sem precedentes, principalmente porque está apresentando bom desempenho na solução de inúmeras tarefas que demandam a análise de dados complexos. Dentre as tarefas solucionadas por algoritmos de AP, podem ser citadas, por exemplo, o reconhecimento de voz (DAHL et al., 2012; SAK et al., 2014), processamento de linguagem natural (WU et al., 2016), medicina diagnóstica (LITJENS et al., 2017; CHENG et al., 2016), jogos (SILVER et al., 2016, 2018), análise do mercado financeiro (HEATON et al., 2017), detecção de fraudes e *malwares* (KNORR, 2015; DAHL et al., 2013), detecção de ataques DDoS (YUAN et al., 2017a; YADAV; SUBRAMANIAN, 2016) e Visão Computacional (KRIZHEVSKY et al., 2012; SZEGEDY et al., 2015a,b; HE et al., 2016; HU et al., 2017).

No campo da Visão Computacional, as Redes Neurais Convolucionais (RNCs) são o estado da arte em reconhecimento e classificação de imagens, e atualmente são utilizadas em diversas tarefas críticas em segurança, como veículos autônomos (BOJARSKI et al., 2016), sistemas de vigilância (DING et al., 2018), reconhecimento biométrico (MIDDLEHURST, 2015; BAE et al., 2018) e identificação de dígitos e assinaturas manuscritas (SRIVASTAVA et al., 2019; TOLOSANA et al., 2018). Entretanto, mesmo apresentando um desempenho superior ao de um humano em tarefas de alta complexidade (veja a Figura 1), trabalhos como Szegedy et al. (2013); Goodfellow et al. (2015); Moosavi-Dezfooli et al. (2016); Papernot et al. (2016a); Carlini e Wagner (2017c) alertam que os algoritmos de aprendizado, em especial aqueles por trás das RNCs, estão suscetíveis a ações maliciosas chamadas de **ataques contraditórios**. Em Visão Computacional, os ataques contraditórios mais comuns buscam adulterar a predição de um algoritmo de classificação por meio de uma **imagem contraditória**. As imagens contraditórias contêm perturbações pequenas e, às vezes imperceptíveis aos olhos humanos, que são geradas por meio de um algoritmo de ataque. Essa vulnerabilidade que as RNCs e outros algoritmos de aprendi-

zado profundo apresentam perante as imagens contraditórias pode resultar em acidentes e prejuízos de larga escala, impossibilitando a utilização desses modelos em tarefas onde a segurança é crucial (KLARREICH, 2016). A Figura 2 ilustra um cenário de ataque contraditório a um sistema de reconhecimento de imagens de um veículo autônomo.

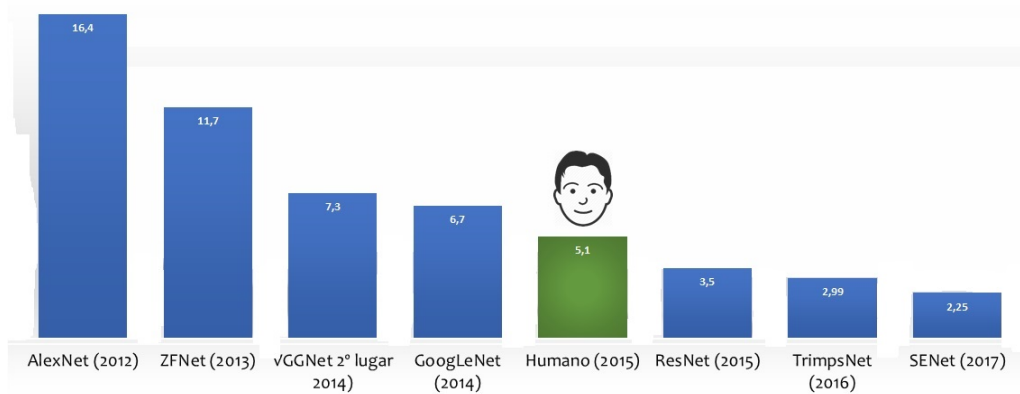


FIG. 1: Desempenho das arquiteturas de RNC vencedoras do desafio *ILSVRC top-5 error rate* (RUSSAKOVSKY et al., 2015) durante os anos. Desde 2015, as RNCs vêm superando o humano neste desafio (KARPATHY, 2014).

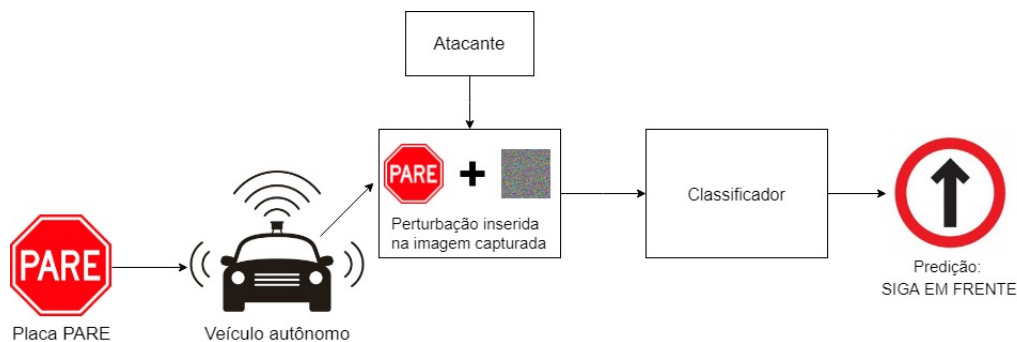


FIG. 2: Um ataque contraditório ao sistema responsável pelo reconhecimento de imagens de um veículo autônomo poderia acarretar em acidentes de trânsito.

1.2 PROBLEMA

Desde o momento em que Szegedy et al. (2013) destacou de maneira inédita as vulnerabilidades dos algoritmos de aprendizado profundo, as atenções de muitos pesquisadores têm se voltado para a elaboração de métodos de defesa contra imagens contraditórias. Apesar de inicialmente apresentarem bons resultados, muitos métodos de defesa, com o passar do tempo, se tornaram ineficazes por (i) não conseguirem se adaptar às novas abordagens de ataque (CHAKRABORTY et al., 2018, pág. 22) e (ii) por utilizarem abordagens determinísticas, *i.e.*, defesas que, a cada execução, utilizam sempre o mesmo procedimento

para detecção, facilitando o entendimento de seu *modus operandi* pelo atacante (MENG; CHEN, 2017, pág. 141).

Como forma de diminuir a dependência de ataques contraditórios específicos e dificultar a previsão do comportamento da defesa pelo atacante, um trabalho recente elaborou um método de detecção promissor, denominado *MagNet* (MENG; CHEN, 2017). O *MagNet* é um método de defesa não determinístico que escolhe aleatoriamente, e em tempo de execução, **um** componente de defesa a ser utilizado no cálculo de um limiar necessário para classificar as imagens como legítimas ou contraditórias. Apesar do *MagNet* ter apresentado um bom desempenho na detecção de imagens contraditórias geradas por diferentes algoritmos de ataque, Carlini e Wagner (2017b) demonstraram que, mesmo com o não determinismo proporcionado pela seleção aleatória de um componente de defesa, o *MagNet* ainda pode ser evadido por imagens contraditórias.

1.3 HIPÓTESE E OBJETIVO

Mesmo com os resultados apresentados pelo *MagNet* em Carlini e Wagner (2017b), o não determinismo ainda pode ser uma alternativa promissora na proteção de classificadores contra imagens contraditórias (VOROBAYCHIK; KANTARCIOGLU, 2018, pág. 133). Portanto, este trabalho levantou a hipótese de que *a seleção aleatória de múltiplos componentes de defesa em tempo de execução pode ampliar o efeito do não determinismo, produzindo, desta forma, um método de defesa mais robusto contra ataques contraditórios.*

Diante do exposto, este trabalho tem como principal objetivo buscar evidências da validade da hipótese levantada, ao propor o *MultiMagNet*. O *MultiMagNet* é uma arquitetura não determinística para detecção e reforma de imagens contraditórias inspirada no *MagNet*. Ao contrário do *MagNet*, que escolhe aleatoriamente, e em tempo de execução, apenas um componente de defesa, o *MultiMagNet* escolhe em tempo de execução múltiplos componentes de defesa como forma de aumentar a diversidade de opções e, consequentemente, ampliar o efeito do não determinismo e a robustez do método de defesa. Pelos resultados obtidos a partir de um estudo comparativo, foi possível validar a hipótese levantada por este trabalho ao mostrar que, em um cenário onde o atacante tem acesso aos parâmetros e à arquitetura do classificador, o *MultiMagNet* é superior ao *MagNet* na proteção contra imagens contraditórias geradas por diferentes algoritmos de ataque.

1.4 CONTRIBUIÇÕES ESPERADAS

Em suma, as principais contribuições esperadas desta dissertação são:

- a melhora da capacidade de proteção de classificadores, em relação aos métodos de defesa estado da arte, com a elaboração do *MultiMagNet*: uma arquitetura não determinística baseada na escolha aleatória de múltiplos componentes de defesa contra imagens contraditórias;
- a realização de um estudo comparativo entre o *MultiMagNet* e o método de defesa *MagNet*.

1.5 ORGANIZAÇÃO DO TRABALHO

O restante desta dissertação está organizado da seguinte forma: o Capítulo 2 introduz os conceitos básicos para o entendimento deste trabalho; o Capítulo 3 faz uma síntese dos principais métodos de defesa contra imagens contraditórias disponíveis na literatura; o Capítulo 4 apresenta em detalhes a arquitetura de defesa do *MultiMagNet*; o Capítulo 5 descreve os experimentos realizados e discute os resultados obtidos; por fim, o Capítulo 6 traz as considerações finais e indica algumas sugestões de trabalhos futuros.

2 CONCEITOS BÁSICOS

2.1 APRENDIZADO DE MÁQUINA

O Aprendizado de Máquina é uma sub-área da Inteligência Artificial que desenvolve algoritmos que buscam automatizar tarefas pela identificação de padrões em um conjunto de dados (FACELI et al., 2011, pág. 2). Basicamente, esses algoritmos podem elaborar um modelo (ou hipótese) para solucionar uma determinada tarefa utilizando três abordagens de aprendizado distintas: (i) aprendizado supervisionado, (ii) aprendizado não-supervisionado e (iii) aprendizado por reforço.

A abordagem de aprendizado supervisionado é comumente empregada para solucionar problemas de (i) regressão e (ii) classificação. Geralmente, em tarefas de regressão, é fornecido um conjunto inicial de dados que contém valores contínuos, que podem ou não estar organizados de forma temporal. Este conjunto de dados é utilizado para o treinamento de um modelo, que posteriormente irá prever um valor contínuo a partir de um determinado conjunto de características fornecido, dentro de uma margem de erro previamente calculada. Já em tarefas de classificação, é fornecido um conjunto inicial de dados rotulados, onde cada rótulo representa a saída esperada que o algoritmo de AM deverá prever. O algoritmo de AM então utiliza a experiência adquirida pelo treinamento nesse conjunto de dados inicial para elaborar um modelo que consiga generalizar a predição dos rótulos de futuros dados de entrada. A Figura 3 ilustra graficamente o processo de aprendizagem de um algoritmo de AM em uma abordagem supervisionada.

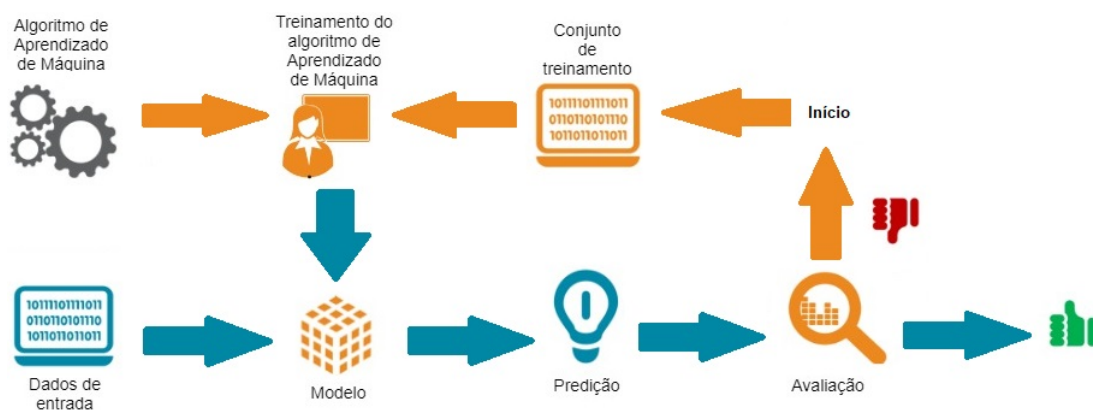


FIG. 3: O processo de aprendizagem de um algoritmo de AM em uma abordagem supervisionada. O principal objetivo dos algoritmos de AM é aprender sobre os dados, *i.e.* identificar padrões ocultos para solucionar automaticamente tarefas sem nenhuma programação *hard-coded* (GOODFELLOW et al., 2016, pág. 99).²

Na abordagem de aprendizado não-supervisionada é utilizado um conjunto de dados não rotulado no processo de aprendizagem. Dentre as tarefas de AM que utilizam a abordagem não supervisionada estão as tarefas de agrupamento (ou *clustering*), regras de associação e redução de dimensionalidade (PAPERNOT, 2018a, pág. 7);

Já a abordagem de aprendizado por reforço tem como objetivo produzir, por tentativa e erro, um modelo que seja capaz de definir automaticamente o comportamento que deve apresentar em um determinado contexto para maximizar o seu desempenho. Algoritmos com essa abordagem de aprendizado são gulosos, pois escolhem as ações que levam a uma maior recompensa. Geralmente são utilizados na solução de tarefas que envolvem robótica e automatização de jogos (PAPERNOT, 2018a, pág. 7).

2.2 APRENDIZADO PROFUNDO

O Aprendizado Profundo é uma sub-área de AM que simula as funções do cérebro humano ao analisar e extrair automaticamente as características contidas em um conjunto de dados, de modo a reduzir a necessidade de intervenção humana e a dependência de especialistas na realização da engenharia de atributos (BROWNLEE, 2016; MAHAPATRA, 2018; GOODFELLOW et al., 2016, pág. 99). Basicamente, os algoritmos de AP aprendem abstrações de alto-nível utilizando uma rede profunda que contém uma estrutura hierárquica formada por inúmeras camadas de processamento, onde são aplicadas várias transformações lineares e não lineares. Essa estrutura hierárquica permite ao algoritmo aprender conceitos mais complexos a partir da construção de conceitos mais simples (GOODFELLOW et al., 2016, págs. 1-2). Segundo Guo et al., (2016, pág. 28), os algoritmos de AP podem ser divididos em cinco categorias principais: (i) Redes Neurais Convolucionais (RNCs), (ii) *Autoencoders*, (iii) Máquinas de *Boltzmann* Restritas, (iv) Codificação Esparsa e (v) Redes de Memória de Longo Prazo (do inglês, *Long Short-Term Memory - LSTM*). As Seções 2.2.1 e 2.2.2 apresentam, respectivamente, mais informações sobre as RNCs e os *autoencoders*. Entretanto, para mais detalhes sobre as Máquinas de *Boltzmann* Restritas, Codificação Esparsa e LSTMs recomenda-se a referência de Goodfellow et al. (2016).

2.2.1 REDES NEURAIAS CONVOLUCIONAIS

As RNCs, também conhecidas como *ConvNets* (LECUN et al., 2010), são um tipo especial de rede neural profunda que atualmente representam o estado da arte na solução de

²Adaptado de <http://blogs.teradata.com/data-points/building-machine-learning-infrastructure-2>. Acessado em 21 de março de 2018.

diversas tarefas de visão computacional. Esses algoritmos de AP utilizam uma representação hierárquica composta por várias funções paramétricas destinadas ao aprendizado de características e processamento de dados em formato matricial, como por exemplo as imagens (PAPERNOT, 2018a, pág. 29); (GOODFELLOW et al., 2016, pág. 330). As RNCs são formadas basicamente por três grupos de camadas principais: (i) camadas de convolução, (ii) camadas de *pooling* (também conhecidas como *downsampling* ou subamostra) e (iii) camadas densamente conectadas. A Figura 4 ilustra um modelo de RNC para classificação de imagens. A seguir, esses grupos de camadas são apresentados com mais detalhes.

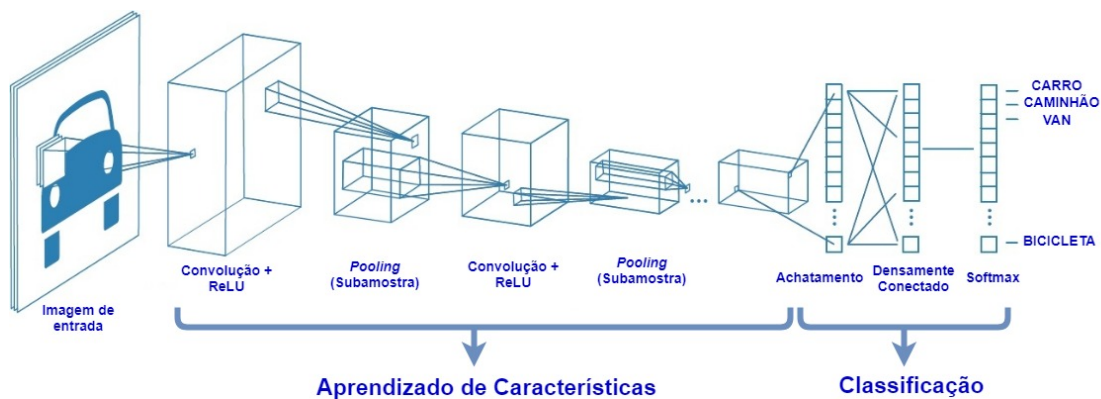


FIG. 4: Exemplo de uma RNC com várias camadas internas. Todo o processo de extração e aprendizado de características é realizado pelas camadas de convolução e *pooling*. Ao final, a camada densamente conectada atua como uma rede neural convencional ao classificar a imagem em cada uma das classes do problema.⁴

2.2.1.1 CAMADA DE CONVOLUÇÃO

As RNCs têm este nome devido às operações de convolução que são realizadas pelas suas camadas internas. O principal objetivo do operador de convolução é extrair características da imagem de entrada, preservando o relacionamento espacial entre os *pixels*. O processo de aprendizagem das características da imagem é realizado utilizando pequenas seções de dados de entrada, onde é aplicado uma matriz conhecida como filtro ou *kernel*. Geralmente, as dimensões da matriz do filtro são menores que as dimensões da imagem, o que significa que são necessários alguns movimentos no filtro (esses movimentos são conhecidos como *strides*) em cada seção da imagem até que o filtro tenha sido aplicado em toda a imagem (veja a Figura 5). Segundo Guo et al. (2016); Liu et al. (2017a); Goodfellow et al. (2016), existem três principais vantagens na operação de convolução: (i) a

⁴Adaptado de <https://www.mathworks.com/discovery/convolutional-neural-network.html>. Acessado em 13 de março de 2018.

redução da imagem de entrada pelo extração das características mais importantes no cálculo de multiplicação de matrizes, (ii) a descoberta de correlações existentes entre *pixels* próximos e a (iii) representação equivariante, que afirma que a rede consegue se tornar invariante a pequenas modificações na imagem, como escala, posição, entre outros. Após cada camada convolucional, é aplicada uma função de ativação cujo objetivo é introduzir a não-linearidade. Essa etapa é chamada de *detector stage* (GOODFELLOW et al., 2016, pág. 339) e a função de ativação mais utilizada é a ReLU (*Rectified Linear Unit*), que zera todos os *pixels* com valores negativos. Formalmente, $ReLU(x) = \max(0, x)$.

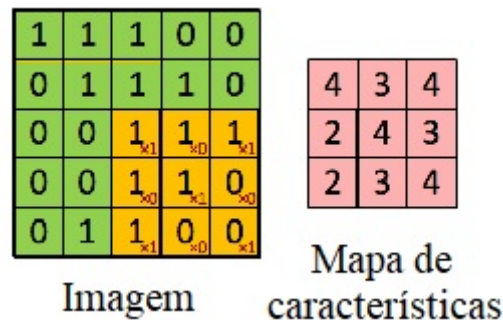


FIG. 5: O processo de convolução feito em uma RNC. O filtro Y , representado pela matriz amarela é aplicado em uma imagem de entrada, representada pela matriz verde G . O resultado é um mapa de características representado pela matriz vermelha R . É importante perceber que neste processo houve 9 *strides*, que representam a dimensão 3x3 do mapa de características. Cada célula da matriz R é calculada pelo produto interno entre a matriz Y e a submatriz resultante do *stride* em G ⁶.

2.2.1.2 CAMADA DE *POOLING*

A camada de *pooling* comumente vem após a camada de convolução e a função de ativação. A principal função da camada de *pooling* é reduzir a dimensionalidade dos mapas de característica para a prevenção de *overfitting* (GUO et al., 2016, pág. 28). Entretanto, dependendo do conjunto de treinamento e do algoritmo de classificação utilizado no processo de aprendizagem, o modelo resultante pode apresentar um problema quanto à sua capacidade de generalização chamado *overfitting*. O *overfitting* ocorre quando um algoritmo de classificação se sobreajusta ao seu conjunto de treinamento e produz um modelo que não tem uma boa capacidade de generalização para inferir, de forma robusta, os rótulos de novas amostras. Quando um modelo apresenta problemas de generalização, como no caso do *overfitting*, todo o processo de aprendizagem deve ser refeito até que o modelo resultante tenha um desempenho satisfatório.

⁶Adaptado de http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution. Acessado em 22 de março de 2018.

De uma forma semelhante à camada de convolução, na camada de *pooling* há a passagem de um filtro, aqui conhecido como janela ou *window* que realiza uma operação (baseado no método de cálculo do *pooling*) em cada seção do mapa de características (*strides*). Os métodos de cálculo mais utilizados são o *max-pooling*, onde é escolhido o maior pixel presente na janela do *pooling* (ver a Figura 6), *mean-pooling*, onde é calculado a média dos valores dos pixels presentes na janela e *sum-pooling*, onde é realizada a soma ao invés da média.

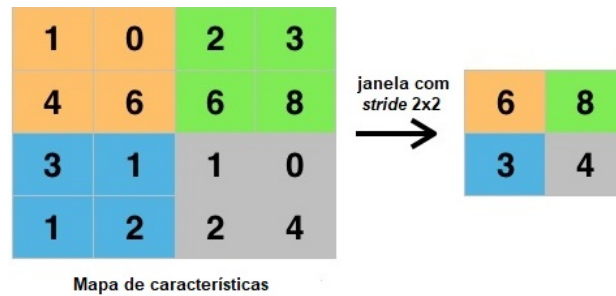


FIG. 6: Exemplo de uma janela de *max-pooling* com *stride* 2×2 ⁷.

2.2.1.3 CAMADA DENSAMENTE CONECTADA

Após a última camada de *pooling*, há um grupo de camadas conhecidas como camadas densamente conectadas. As camadas densamente conectadas são formadas por neurônios artificiais que operam de maneira semelhante às redes *multilayer perceptrons*, e representam cerca de 90% dos parâmetros da RNC. Esse grupo de camadas fica responsável por produzir as respostas do modelo para uma determinada imagem de entrada, após ter sido calibrado em um conjunto de treinamento por um algoritmo chamado de *Backpropagation* (RUMELHART et al., 1988).

Durante o aprendizado das RNCs, o *Backpropagation* primeiramente calcula o erro de predição do modelo, ao comparar as saídas probabilísticas *softmax*⁸ \tilde{y}_i do modelo com os rótulos y_i que representam a classe de cada imagem. Esse erro geralmente é obtido pela função de custo MSE (*Mean Squared Error*), como mostrado pela Equação 1.

$$MSE = \frac{1}{n} \sum_{n=1}^n (y_i - \tilde{y}_i)^2 \quad (1)$$

⁷Adaptado de https://en.wikipedia.org/wiki/File:Max_pooling.png. Acessado em 02 de fevereiro de 2019.

⁸A função *softmax*, definida pela Equação 10, produz como saída um vetor de probabilidades \vec{v} com n elementos, sendo, $\sum_{i=1}^n v_i = 1$, onde cada elemento v_i corresponde à probabilidade da imagem de entrada pertencer à classe representada pelo rótulo y_i .

Após o cálculo da função de custo MSE, o *Backpropagation* utiliza um algoritmo de otimização chamado Gradiente Descendente que, de forma recursiva, retropropaga os ajustes, necessários para minimizar a função de custo, aos parâmetros das subcamadas anteriores, que juntas formam a camada densamente conectada da RNC. Essa queda da função de custo é geometricamente representada pela convergência dos parâmetros a um mínimo local no espaço multidimensional. Contudo, devido à grande quantidade de parâmetros que as RNCs possuem e à necessidade de se utilizar milhares de amostras para treinamento, problemas de lentidão no processo de aprendizagem e *overfitting* são comuns. Por isso, algoritmos auxiliares como o Gradiente Descendente Estocástico (GDE) e *Dropout* são frequentemente utilizados durante o processo de aprendizagem.

O *Dropout* (SRIVASTAVA et al., 2014) é uma técnica para treinamento de algoritmos que anula, de maneira aleatória e com uma probabilidade p , alguns neurônios e suas ligações durante a fase de treinamento para mitigar os efeitos do *overfitting* e diminuir o tempo necessário para treinamento dos modelos. Já o GDE é uma variação do algoritmo de gradiente descendente que, iterativamente, atualiza os parâmetros da camada densamente conectada para cada amostra de treinamento. É frequentemente utilizado por convergir mais rapidamente a um mínimo local que outras variações do gradiente descendente mais tradicionais.

2.2.2 AUTOENCODERS

Os *autoencoders* são redes neurais profundas treinadas para gerar como saída uma reconstrução \hat{x} a partir de uma imagem de entrada x (GOODFELLOW et al., 2016, pág. 502). Formalmente, um *autoencoder* $ae = c \circ d$ contém dois componentes: um codificador $c : \mathbb{S} \rightarrow \mathbb{H}$, e um decodificador $d : \mathbb{H} \rightarrow \hat{\mathbb{S}}$, onde \mathbb{S} é o espaço de entrada, \mathbb{H} é o espaço da representação comprimida aprendida pelo *autoencoder* e $\hat{\mathbb{S}}$ representa o espaço de entrada \mathbb{S} reconstruído. O erro de reconstrução $ER_{ae(x)}$ geralmente é utilizado como função de custo do *autoencoder* e é definido de acordo com a Equação 2:

$$ER_{ae(x)} = \|x - ae(x)\|_p \quad (2)$$

Na Equação 2, x é a imagem de entrada, $ae(x)$ é a reconstrução de x feita pelo *autoencoder* e p é a métrica de distância (sobre as métricas de distância, consulte a Seção 2.3.1.3). *Autoencoders* geralmente são utilizados para redução de dimensionalidade e aprendizado de características, uma vez que priorizam as propriedades mais importantes das imagens (GOODFELLOW et al., 2016, pág. 502). A Figura 7 ilustra graficamente um exemplo de *autoencoder*.

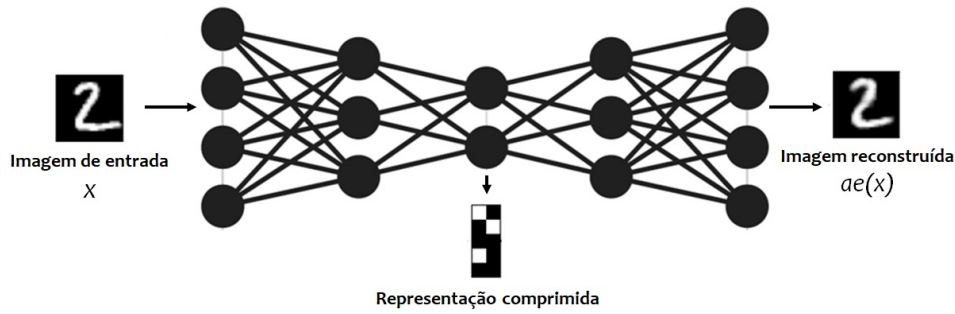


FIG. 7: Um exemplo de *autoencoder*. Ao receber uma imagem de entrada, o *autoencoder* a reconstrói a partir da representação comprimida aprendida pela etapa de codificação¹⁰.

Nos experimentos realizados neste trabalho foram utilizados dois tipos de *autoencoders*: os *autoencoders* convolucionais, e os *denoising autoencoders*. Ambas as arquiteturas de *autoencoders* são semelhantes à arquitetura das Redes Neurais Convolucionais, entretanto, os *denoising autoencoders* inserem um ruído¹¹ intencional nas imagens de treinamento. Esta medida evita que, após o treinamento, os *autoencoders* se transformem em funções-identidade.

2.3 IMAGENS CONTRADITÓRIAS

Uma imagem contraditória¹² é uma imagem que contém pequenas perturbações, geradas por meio de um algoritmo de ataque contraditório, com o objetivo de causar erros aos modelos de classificação (veja a Figura 8). Formalmente, sendo F um classificador treinado com imagens legítimas (*i.e.* imagens que não possuem perturbações maliciosas) e x uma imagem legítima (onde $x \in \mathbb{R}^{w \times h \times c}$ tal que w e h são as dimensões da imagem e c a quantidade de canais de cor), é gerada, a partir de x , uma imagem x' , tal que $x' = x + \delta x$, onde δx é a perturbação necessária para que a imagem x cruze a fronteira de decisão, tal que $F(x) \neq F(x')$. A Figura 9 ilustra graficamente o efeito resultante da inserção da perturbação δx na imagem x . O ato de gerar imagens contraditórias e enviá-las aos modelos de classificação com o objetivo de induzi-los ao erro é conhecido como **ataque contraditório**. De acordo com Cao e Gong (2017, pág. 281), uma imagem contraditória é satisfatória se atender a dois requisitos: (i) se as perturbações contidas nessa imagem

¹⁰Adaptado de <https://www.curso-r.com/blog/2017-06-26-construindo-autoencoders/>. Acessado em 02 de fevereiro de 2019.

¹¹É importante ressaltar que neste trabalho, os termos **ruído** e **perturbação** designam formas diferentes de corrompimento em uma imagem. O ruído é qualquer distorção já naturalmente presente ou gerada em uma imagem de maneira não sistemática, *i.e.* sem utilizar um algoritmo malicioso. Uma perturbação é uma distorção sistemática gerada maliciosamente por um algoritmo de ataque, produzindo uma imagem contraditória. Para mais detalhes, veja a Seção 2.3.1.2.

¹²O termo em português "imagem contraditória" foi traduzido do inglês ("*adversarial example*") e cunhado pelos autores deste trabalho.

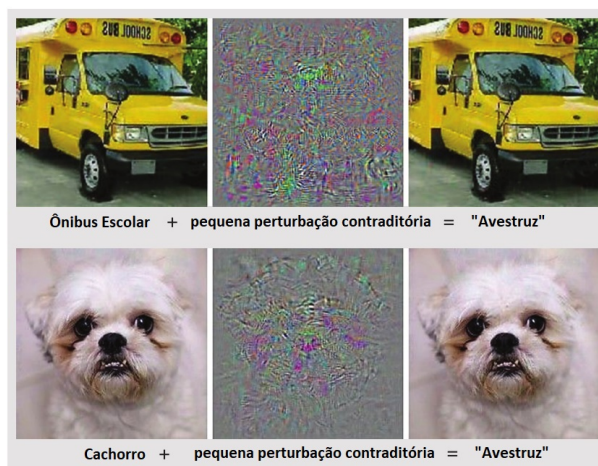


FIG. 8: Perturbações imperceptíveis, inseridas maliciosamente em uma imagem, podem induzir um modelo de classificação ao erro. Adaptado de Klarreich (2016).

forem imperceptíveis aos olhos humanos e (ii) se consegue produzir uma saída incorreta em um modelo classificatório.

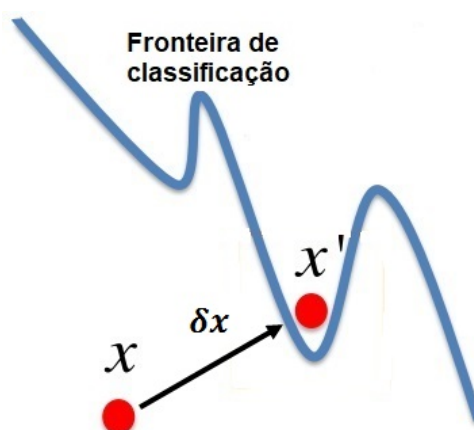


FIG. 9: O objetivo de um ataque contraditório é gerar uma perturbação δx e inserí-la em uma imagem x a ponto de fazer com que x cruze a fronteira de decisão, e desta forma, produza um erro de classificação. Adaptado de Cao e Gong, (2017, pág. 282).

As vulnerabilidades que as RNCs e outros algoritmos de AP apresentam diante de ataques contraditórios são uma consequência do Efeito *Clever Hans*. O Efeito *Clever Hans* é um termo originalmente cunhado por psicólogos alemães no início do século XX como referência ao comportamento atípico de um cavalo chamado *Hans*. Na época, o cavalo *Hans* aparentava possuir certas habilidades intelectuais ao bater seu casco no chão o número de vezes correspondente às respostas de várias perguntas de matemática que eram feitas a ele em público. Entretanto, após a realização de diversos experimentos conduzidos com o cavalo *Hans*, psicólogos chegaram à conclusão de que, na verdade, *Hans* não sabia fazer cálculos matemáticos, mas que, de alguma forma, ele conseguiu

identificar sinais de quando deveria parar de bater o seu casco no chão ao observar o comportamento da plateia e do seu dono, Wilhelm von Osten.

De maneira semelhante ao cavalo *Hans*, os modelos de AP conseguem produzir respostas corretas para problemas complexos envolvendo o reconhecimento e classificação de imagens, sem realmente aprender sobre essas imagens, o que os tornam suscetíveis a ataques contraditórios (GERSHGORN, 2016; KUMAR; MEHTA, 2017).

2.3.1 TAXONOMIA DAS IMAGENS CONTRADITÓRIAS

Esta seção se baseia nos trabalhos de Barreno et al. (2010); Huang et al. (2011); Yuan et al. (2017b); Kumar e Mehta (2017); Xiao (2017); Brendel et al. (2017) para propor uma taxonomia de imagens contraditórias categorizada em 3 diferentes eixos, como mostrado pela Figura 10: (i) escopo da perturbação, (ii) visibilidade da perturbação e (iii) medida da perturbação. Os ramos destacados em azul representam os tipos de imagens contraditórias que foram utilizados nos experimentos detalhados no Capítulo 5.

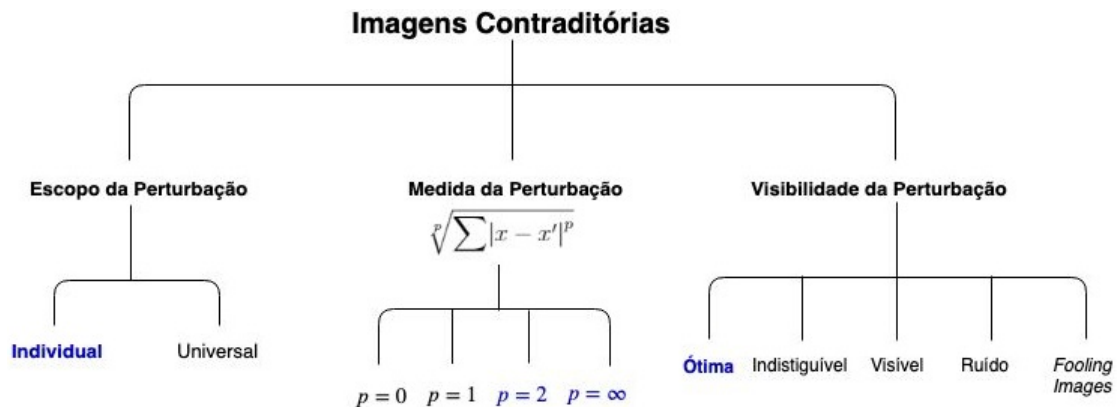


FIG. 10: Taxonomia proposta para as imagens contraditórias.

2.3.1.1 ESCOPO DA PERTURBAÇÃO

As imagens contraditórias podem conter perturbações com escopo individual e universal:

- **perturbações com escopo individual:** é o tipo de escopo mais comum, onde as perturbações são geradas individualmente para cada imagem de entrada;
- **perturbações com escopo universal:** é gerada uma única perturbação que, ao ser inserida em qualquer imagem de entrada, pode causar erros de classificação em modelos (METZEN et al., 2017b; MOOSAVI-DEZFOOLI et al., 2017).

2.3.1.2 VISIBILIDADE DA PERTURBAÇÃO

A eficiência e visibilidade das perturbações geradas por algoritmos de ataque e inseridas em imagens podem ser categorizadas em:

- **perturbação ótima:** perturbações ótimas conseguem causar erros de classificação nos modelos de aprendizado e, ao mesmo tempo, serem imperceptíveis aos olhos humanos;
- **perturbação indistinguível:** perturbações indistinguíveis também não são perceptíveis aos olhos humanos, entretanto não conseguem causar erros de classificação em modelos de AP;
- **perturbação visível:** perturbações visíveis conseguem enganar modelos de AP, porém podem ser facilmente identificadas por humanos (KARMON et al., 2018; BROWN et al., 2017; EVTIMOV et al., 2017);
- **ruído:** ao contrário das perturbações, os ruídos são quaisquer variações de natureza não maliciosa que podem estar presentes em uma imagem de entrada;
- **fooling images:** são perturbações que corrompem as imagens a um ponto de torná-las irreconhecíveis aos humanos. Mesmo assim, os modelos de classificação acreditam que essas imagens corrompidas pertencem a uma das classes do problema, atribuindo, às vezes, uma alta taxa de confiança na predição (NGUYEN et al., 2015). Essas imagens corrompidas também são conhecidas como *rubbish class examples* (GOODFELLOW et al., 2015, pág. 10).

2.3.1.3 MEDIDA DA PERTURBAÇÃO

Devido às dificuldades de se definir uma métrica que meça a capacidade de visão humana, são utilizadas as p-normas para medir as perturbações adicionadas às imagens legítimas (MENG; CHEN, 2017, pág. 137). A p-norma L_p calcula a distância $\|x - x'\|_p$, onde $p \in \{0, 1, 2, \infty\}$. A p-norma é definida de acordo com a Equação 3:

$$L_p = \sqrt[p]{\sum |x - x'|^p} \quad (3)$$

A Equação 3 se aplica quando $p = 1$ (distância de *Manhattan*) e $p = 2$ (distância euclidiana). Quando $p = 0$, é contado o número de *pixels* que foram modificados em uma imagem legítima para gerar uma imagem contraditória. A distância L_∞ mede a diferença máxima de todos os *pixels* nas posições correspondentes entre duas imagens.

Na norma L_∞ , é permitido que cada *pixel* seja alterado dentro de um limite máximo de perturbação, sem que haja nenhuma restrição para o número de *pixels* que podem ser alterados. Formalmente, $L_\infty = \|x - x'\|_\infty = \max(|x_1 - x'_1|, |x_2 - x'_2|, \dots, |x_n - x'_n|)$.

2.3.2 TAXONOMIA DOS ATAQUES CONTRADITÓRIOS

Essa seção também se baseia nos conceitos e definições de trabalhos como Barreno et al. (2010); Yuan et al. (2017b); Kumar e Mehta (2017); Xiao (2017); Brendel et al. (2017); Akhtar e Mian (2018) para propor uma nova taxonomia que organiza os ataques contraditórios em cinco diferentes eixos, como mostrado pela Figura 11: (i) influência do atacante, (ii) conhecimento do atacante, (iii) especificidade do ataque, (iv) cálculo do ataque e (v) abordagem de ataque. Os ramos destacados em azul representam os tipos de ataques realizados para os experimentos detalhados no Capítulo 5.

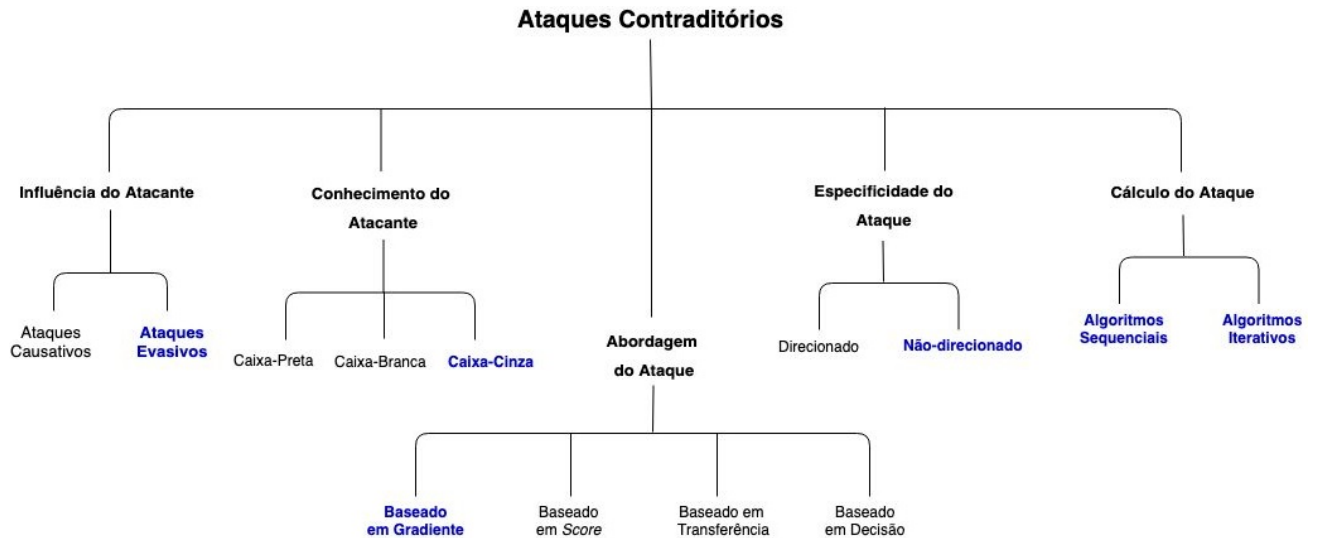


FIG. 11: Taxonomia proposta para os ataques contraditórios.

2.3.2.1 INFLUÊNCIA DO ATACANTE

A influência de um atacante define como ele controlará o processo de aprendizado dos modelos de AP. De acordo com Xiao (2017, pág. 27), o atacante pode realizar dois tipos de ataque, levando em consideração a sua influência sobre o modelo de classificação: (i) ataques causativos ou de envenenamento e (ii) ataques evasivos ou exploratórios. A Figura 12 ilustra como o modelo de aprendizado é influenciado em cada um dos dois tipos de ataque.

- **Ataques causativos ou de envenenamento:** nos ataques de envenenamento, o atacante tem influência sobre o modelo de AP durante a fase de treinamento. Nesse

tipo de ataque, as amostras do conjunto de treinamento são poluídas ou adulteradas para que seja produzido um modelo de AP incompatível com a distribuição original dos dados;

- **ataques evasivos ou exploratórios:** ao contrário dos ataques causativos, nos ataques evasivos o atacante tem influência sobre os modelos de AP durante a fase de inferência (ou teste). É o tipo de ataque mais comum, onde o atacante gera imagens contraditórias que produzem erros de classificação em modelos de AP com uma alta taxa de confiança. Ataques evasivos também podem ter caráter exploratório, onde o objetivo é coletar informações sobre o modelo de classificação, como parâmetros, arquitetura, funções de custo, *etc.* O ataque exploratório mais comum é do tipo *input/output*, onde o atacante fornece amostras elaboradas por ele ao modelo e observa suas respectivas saídas, buscando reproduzir um modelo substituto, similar ao modelo observado. O ataque *input/output* é, geralmente, a primeira etapa para realizar um ataque de caixa-preta (veja a Seção 2.3.2.2).

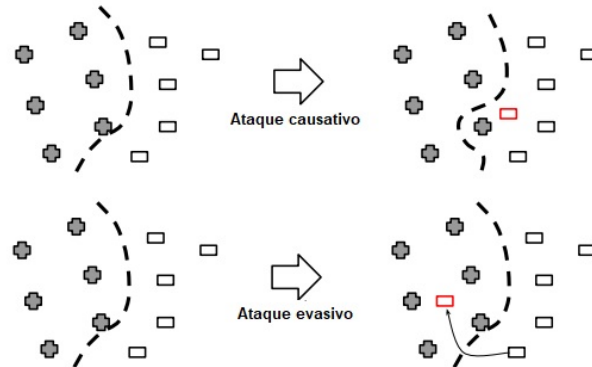


FIG. 12: Ataques com influência causativa e evasiva em um modelo de classificação binário. Os ataques causativos influenciam o modelo durante sua fase de treinamento, o que altera a fronteira de decisão final. Por outro lado, os ataques com influência evasiva almejam encontrar um ponto crítico no espaço de dados de modo que uma amostra contraditória gerada consiga evadir a fronteira de decisão sem ser detectada. Por serem conduzidos durante a fase de inferência, os ataques evasivos não alteram a fronteira de decisão dos modelos de AP. Adaptado de Xiao (2017, pág. 27).

2.3.2.2 CONHECIMENTO DO ATACANTE

Esse eixo também é conhecido como modelo de ataque ou *threat model*. Um atacante pode realizar três diferentes ataques contraditórios de acordo com a quantidade de informação sobre o modelo que está à sua disposição: (i) ataques de caixa-branca, (ii) ataques de caixa-preta e (iii) ataques de caixa-cinza.

- **Ataques de caixa-branca:** em ataques de caixa-branca, o atacante tem conhecimento e acesso total aos algoritmos, aos parâmetros e às arquiteturas do modelo e do método de defesa, caso haja um presente. Esse cenário de ataque seria o menos frequente em aplicações reais devido à adoção de medidas de proteção (como o controle de usuários, por exemplo) para prevenir o acesso indevido de pessoas não autorizadas aos componentes do sistema. Entretanto, uma vez que os cenários de caixa-branca possibilitam a realização de ataques mais fortes, são utilizados como uma forma de avaliação da robustez dos modelos classificatórios e/ou dos métodos de defesas quando esses são submetidos ao limite. Até o presente momento, elaborar modelos de classificação e/ou métodos resistentes aos ataques de caixa-branca ainda é um problema em aberto.
- **ataques de caixa-preta:** neste cenário, o atacante não tem acesso e nem conhecimento sobre nenhuma informação acerca do modelo de classificação e do método de defesa, caso haja um presente. Cenários de caixa-preta impõem maiores dificuldades ao atacante, contudo são importantes porque reproduzem ataques externos aos modelos quando estão em funcionamento em algum cenário de produção (PAPERNOT et al., 2017). Apesar da maior dificuldade do ataque de caixa-preta, o atacante ainda consegue evadir modelos com imagens contraditórias devido à **transferibilidade das imagens contraditórias**. Trabalhos como o de Szegedy et al. (2013) e Papernot et al. (2017) mostraram que uma imagem contraditória gerada em um classificador é capaz de se transferir para outros modelos de classificação, inclusive modelos criados com diferentes algoritmos de AM. Com esta propriedade a favor do atacante, ele pode, com um ataque causativo, criar um modelo empírico chamado de substituto (ou de *surrogate model*) que possua parâmetros semelhantes ao modelo que ele deseja evadir. Com o modelo substituto, o atacante então pode utilizá-lo para gerar imagens contraditórias e enviá-las ao modelo de classificação almejado;
- **ataques de caixa-cinza:** este cenário de ataque foi proposto primeiramente por Meng e Chen (2017). Em cenários de ataque de caixa-cinza, o atacante tem acesso ao modelo de classificação, mas dispõe de nenhuma informação sobre o método de defesa. Os ataques de caixa-cinza são uma alternativa intermediária para a avaliação de métodos de defesa e modelos de classificação, pois oferecem uma maior ameaça, se comparada aos cenários de ataque de caixa-preta, sem dar uma ampla vantagem ao atacante ao fornecer todas as informações sobre o método de defesa e modelo, como é feito no ataque de caixa-branca.

2.3.2.3 ESPECIFICIDADE DO ATAQUE

No que diz respeito à especificidade do ataque, um ataque contraditório pode ser **direcionado**, quando é gerada uma imagem contraditória de modo que o modelo a classifique em uma determinada classe de interesse do atacante, ou **não-direcionado**, quando é gerada uma imagem contraditória que cause um erro de classificação no modelo, sem que o atacante almeje uma classe específica. Formalmente, sendo x uma imagem legítima, y a classe que x pertence e F um classificador, é produzida a partir de x uma imagem contraditória $x' = x + \delta x$. Em um ataque direcionado, o atacante busca gerar uma perturbação δx que produza como saída em F uma classe específica y' , tal que $F(x + \delta x) = y'$ e $y' \neq y$. Já em um ataque não-direcionado, é gerada uma imagem contraditória x' , tal que $F(x) \neq F(x')$. O ataque direcionado geralmente apresenta um custo computacional maior que o ataque não-direcionado.

2.3.2.4 CÁLCULO DO ATAQUE

As imagens contraditórias podem ser geradas por dois tipos de algoritmos que realizam o cálculo de perturbações: os **algoritmos sequenciais** e os **algoritmos iterativos**. Os algoritmos sequenciais calculam em apenas uma iteração a perturbação que será inserida em uma imagem de entrada. Algoritmos iterativos já necessitam de mais iterações para gerarem uma perturbação. Por utilizarem mais iterações no cálculo de perturbações, os algoritmos iterativos são mais custosos computacionalmente que os algoritmos sequenciais, entretanto as perturbações geradas pelos algoritmos iterativos são menores e mais eficientes em enganarem os modelos de classificação que as perturbações geradas pelos algoritmos sequenciais.

2.3.2.5 ABORDAGEM DO ATAQUE

Os ataques contraditórios também podem ser categorizados de acordo com a abordagem utilizada pelo algoritmo de ataque para a geração de perturbações. De acordo com Brendel et al. (2017), os ataques contraditórios podem ser categorizados em (i) ataques baseados em gradiente, (ii) ataques baseados em *score*, (iii) ataques baseados em transferência e (iv) ataques baseados em decisão.

- **Ataques baseados em gradiente:** esta abordagem é a mais utilizada na literatura, onde os algoritmos de ataque se utilizam de informações detalhadas sobre o modelo, incluindo as informações do gradiente com respeito à entrada. Esta abordagem de ataque é aplicada quando o atacante possui um conhecimento total do

modelo almejado (ataques de caixa-branca);

- **ataques baseados em *score***: os algoritmos de ataque que adotam esta abordagem dependem somente dos *scores* preditos por um modelo, com respeito a uma imagem de entrada, para estimarem um gradiente. Os *scores* mais utilizados são as saídas probabilísticas da última camada da rede profunda (*softmax*) ou da penúltima camada da rede profunda, comumente chamada *logits*, de um modelo almejado. Esta abordagem de ataque é mais utilizada em ataques de caixa-preta;
- **ataques baseados em transferência**: ataques baseados em transferência necessitam apenas do acesso ao conjunto de dados utilizado pelo modelo de classificação almejado. Esse conjunto de dados serve para treinar um modelo auxiliar (*surrogate model*) que gerará as perturbações que serão inseridas nas imagens legítimas. Esta abordagem de ataque é muito útil em cenários de ataque de caixa-preta;
- **ataques baseados em decisão**: esta abordagem utilizada em Brendel et al. (2017) é considerada pelos autores como (i) uma abordagem de ataque mais simples, (ii) mais flexível, (iii) que requer poucas alterações de parâmetros e (iv) competitiva quando comparada com as abordagens de ataque baseados em gradiente. Ataques que utilizam esta abordagem calculam iterativamente uma perturbação utilizando um processo de rejeição de amostras (*rejection sampling*) para que perturbações ainda menores sejam produzidas, e que quando inseridas em uma imagem, torne-a contraditória.

2.3.3 ALGORITMOS DE ATAQUES CONTRADITÓRIOS

Os algoritmos de ataques contraditórios, também conhecidos como métodos de geração de imagens contraditórias, ou simplesmente algoritmos de ataque, são algoritmos de otimização maliciosos que geram e inserem perturbações em imagens legítimas com a finalidade de enganar classificadores. Esses algoritmos de ataque exploram as falhas de generalização dos modelos de AP para produzirem as imagens contraditórias (o Efeito *Clever Hans*. Veja a Seção 2.3).

A Tabela 1 organiza, de acordo com as taxonomias apresentadas nas Seções 2.3.1 e 2.3.2, alguns algoritmos de ataque, como o FGSM (GOODFELLOW et al., 2015), JSMA (PAPERNOT et al., 2016b), L-BFGS (SZEGEDY et al., 2013), *DeepFool* (MOOSAVI-DEZFOOLI et al., 2016), LaVAN (KARMON et al., 2018), *Boundary Attack* (BRENDDEL et al., 2017), *C&W Attack* (CARLINI; WAGNER, 2017c), BIM (KURAKIN et al., 2016a), *Zeroth Order Optimization* (CHEN et al., 2017b), *Projected Gradient Descent* (MADRY

et al., 2017), *Houdini* (CISSE et al., 2017) e ATNs (BALUJA; FISCHER, 2017)¹³. Entretanto, apenas os algoritmos de ataque FGSM, BIM, *DeepFool* e CW serão descritos com mais detalhes pelas próximas seções, principalmente pela relevância que têm em vários trabalhos relacionados presentes na literatura.

2.3.3.1 FAST GRADIENT SIGN METHOD

O *Fast Gradient Sign Method*, ou FGSM, é um algoritmo não-iterativo proposto por Goodfellow et al. (2015), cuja principal característica é a sua complexidade linear, que apesar de torná-lo computacionalmente eficiente, acaba gerando perturbações maiores que as perturbações produzidas por algoritmos iterativos. Dada uma imagem $x \in \mathbb{R}^{w \times h \times c}$, o FGSM gera uma imagem contraditória x' por meio da Equação 4:

$$x' = x - \epsilon \cdot \text{sign}(\vec{\nabla}_x J(\Theta, x, y)) \quad (4)$$

onde $\vec{\nabla}_x$ representa o vetor gradiente, Θ representa os parâmetros da rede, y a classe associada a x , ϵ a quantidade máxima de perturbação que pode ser inserida na imagem x e $J(\Theta, x, y)$ a função de custo utilizada para treinamento da rede.

2.3.3.2 BASIC ITERATIVE METHOD

O *Basic Iterative Method*, ou BIM, é uma versão iterativa do FGSM e foi proposto em Kurakin et al. (2016a). Ao invés de executar um passo de tamanho ϵ na direção do gradiente descendente, como é feito pelo FGSM, vários passos α menores são executados e o resultado é limitado por ϵ , que atua como um teto para que a perturbação não exceda a quantidade desejada pelo atacante. Formalmente, o BIM é um método recursivo, que gera x' segundo a Equação 5:

$$x' = \begin{cases} x'_0 = 0 \\ x'_i = x'_{i-1} - \text{clip}(\alpha \cdot \text{sign} \vec{\nabla}_x J(\Theta, x'_{i-1}, y)) \end{cases} \quad (5)$$

Vale ressaltar que *clip* é um procedimento da linguagem *Python* que organiza os valores fornecidos como parâmetros no intervalo $[0, 1]$.

¹³A categoria **influência do atacante** não está presente na Tabela 1 porque todos os algoritmos de ataque abordados têm influência evasiva ou exploratória.

2.3.3.3 DEEPFOOL

A ideia do *Deepfool*, proposta em Moosavi-Dezfooli et al. (2016) consiste em encontrar a imagem legítima x que esteja mais próxima da fronteira de decisão no espaço e atravessá-la, ou seja, perturbá-la sutilmente para que engane o classificador. Devido à alta dimensionalidade da imagem, é adotada uma abordagem iterativa por aproximação linear. A cada iteração, o *Deepfool* lineariza o modelo em torno do x' intermediário e calcula uma direção de atualização ótima no modelo linearizado. Em seguida, x' é atualizada nessa direção por um pequeno passo α .

2.3.3.4 CARLINI & WAGNER ATTACK

O algoritmo *Carlini & Wagner Attack*, abreviado como C&W ou simplesmente CW, foi proposto por Carlini e Wagner (2017c) e representa o estado da arte na geração de imagens contraditórias. Formalmente, o CW é um ataque iterativo em que, dada uma RNC F com a penúltima camada (*logits*) Z e uma imagem x pertencente à classe t , o ataque utiliza o gradiente descendente para resolver a Equação 6:

$$\text{minimizar } \|x - x'\|_2^2 + c \cdot \ell(x') \quad (6)$$

onde, para x , o ataque procura por uma perturbação $\delta_x = x - x'$ que seja pequena e, ao mesmo tempo, consiga enganar o classificador. Para isso, o hiperparâmetro c é utilizado como uma forma de calcular a quantidade de perturbação mínima necessária para isso. Além de c , há a função de custo $\ell(x')$, que é definida pela Equação 7.

$$\ell(x') = \max(\max\{Z(x')_i : i \neq t\} - Z(x')_t, -conf) \quad (7)$$

Na Equação 7, o hiperparâmetro *conf* se refere à taxa de confiança do ataque. Quanto maior o valor de *conf*, maior é a capacidade de a imagem contraditória enganar os classificadores com uma alta taxa de confiança, *i.e.* apresentar uma probabilidade próxima de 100% em uma classe incorreta. Entretanto, para o tal, são necessárias maiores quantidades de perturbação, que podem ser mais facilmente perceptíveis por humanos.

TAB. 1: Os principais algoritmos de ataques contraditórios.

Algoritmo e referência	Conhecimento do Atacante	Especificidade do Ataque	Escopo da Perturbação	Cálculo do Ataque	Medida da Perturbação	Visibilidade da Perturbação	Abordagem do Ataque
FGSM (32)	caixa-branca	não-direcionado	individual	sequencial	L_∞	ótima	gradiente
JSMA (75)	caixa-branca	direcionado	individual	iterativo	L_0	ótima	gradiente
L-BFGS (91)	caixa-branca	direcionado	individual	iterativo	L_∞	ótima	gradiente
<i>DeepFool</i> (68)	caixa-branca	não-direcionado	universal	iterativo	L_1, L_2, L_∞	ótima	gradiente
LaVAN (45)	caixa-branca	direcionado	universal	iterativo	L_2	visível	gradiente
<i>Boundary Attack</i> (8)	caixa-preta	não-direcionado direcionado	individual	iterativo	L_2	ótima	decisão
C&W <i>Attack</i> (14)	caixa-branca	não-direcionado direcionado	individual	iterativo	L_0, L_2, L_∞	ótima	gradiente
BIM (53)	caixa-branca	não-direcionado	individual	iterativo	L_∞	ótima	gradiente
<i>Zeroth Order Optimization (ZOO)</i> (18)	caixa-preta	não-direcionado direcionado	individual	iterativo	L_2	ótima	<i>score</i>
<i>Projected Gradient Descent (PGD)</i> (61)	caixa-branca	direcionado	individual	iterativo	L_1, L_∞	ótima	gradiente
<i>Houdini</i> (20)	caixa-preta	direcionado	individual	iterativo	L_2, L_∞	ótima	gradiente
ATNs (4)	caixa-branca	direcionado	individual	iterativo	L_∞	ótima	gradiente

2.4 DIVERGÊNCIA DE *JENSEN-SHANNON*

A divergência de *Jensen-Shannon* (JSD) é baseada na métrica de *Kullback-Leibler* (KL). Ao contrário da métrica KL, o JSD calcula de forma simétrica e finita a similaridade entre duas distribuições probabilísticas P e Q que, neste trabalho, representam as saídas da camada *softmax* F de uma RNC, onde dadas duas imagens x e y , $P = F(x)$ e $Q = F(y)$. $P(i)$ e $Q(i)$ indicam, respectivamente, as probabilidades das imagens x e y pertencerem à classe i . O valor da métrica JSD para duas imagens x e y é dado pela Equação 8:

$$JSD(P || Q) = \frac{1}{2}D_{KL}(P || M) + \frac{1}{2}D_{KL}(Q || M) \quad (8)$$

onde

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

O intuito da métrica JSD neste trabalho é identificar o efeito malicioso que perturbações contraditórias ainda menores, inseridas em uma imagem de entrada, podem produzir em um modelo de classificação, uma vez que leva em consideração a distribuição das saídas probabilísticas do classificador no processo de detecção.

3 TRABALHOS RELACIONADOS

3.1 CONSIDERAÇÕES INICIAIS

A ameaça das imagens contraditórias incentivou a comunidade científica a elaborar diversos métodos para a defesa de modelos de classificação. Segundo os autores Yuan et al. (2017b); He et al. (2017); Carlini e Wagner (2017a); Akhtar e Mian (2018), os métodos de defesa contra imagens contraditórias podem ser categorizados em duas diferentes abordagens: **(i) defesas proativas** e **(ii) defesas reativas**. As defesas proativas almejam tornar os modelos de classificação mais robustos às imagens contraditórias. Um modelo é considerado robusto se ele classificar corretamente uma imagem contraditória, como se esta fosse uma imagem legítima. Em contrapartida, as defesas reativas são voltadas para detecção de imagens contraditórias. Essas defesas atuam como um filtro, que identifica as imagens contraditórias antes de alcançarem o classificador da aplicação. As imagens detectadas como contraditórias podem ser descartadas ou enviadas para algum procedimento de recuperação.

A seguir, os principais métodos de defesa proativos e reativos são organizados com base em uma nova taxonomia, inicialmente proposta por esta dissertação, formada por seis diferentes eixos: (i) Métodos Baseados em *Gradient Masking*, (ii) Métodos Baseados em Comitês de Classificadores, (iii) Modelos Auxiliares Detectores, (iv) Métodos Baseados em Abordagens Estatísticas, (v) Métodos Baseados em Pré-Processamento e, por fim, (vi) Métodos Baseados em Proximidade.

3.2 MÉTODOS BASEADOS EM *GRADIENT MASKING*

Métodos de defesa baseados em *Gradient Masking* produzem modelos com gradientes mais suaves, o que dificulta aos algoritmos de ataque baseados em gradiente em encontrar as melhores direções no espaço de uma imagem para realizar a perturbação (GOOD-FELLOW; PAPERNOT, 2017). Basicamente, há na literatura dois métodos de defesa principais que utilizam o conceito de *Gradient Masking*: (i) *Adversarial Training* e (ii) Destilação Defensiva. Ambos os métodos são explicados a seguir.

O *Adversarial Training*, também conhecido como otimização robusta (MADRY et al., 2017), é um método de defesa proativo de força bruta. Basicamente, o *Adversarial Training* tem como objetivo aumentar a robustez de um modelo de classificação treinando-

o em um conjunto de dados contendo imagens legítimas e contraditórias. Formalmente, dada uma tupla $X = (x, y)$, onde x é uma imagem legítima, y a classe que a imagem x pertence e T um conjunto de treinamento contendo apenas a tupla X ¹⁴, tal que $T = \{X\}$, é gerada uma imagem contraditória x' a partir de um algoritmo de ataque A , formando uma nova tupla X' que receberá o mesmo rótulo y da imagem legítima x , tal que $X' = \{x', y\}, x' = A(x)$. Após, o conjunto de treinamento T é enriquecido com a inserção da nova tupla X' e passa a conter duas imagens: $T = \{X, X'\}$. O algoritmo de AP é então treinado no conjunto de treinamento T , resultando em um modelo mais robusto (ver Figura 13).

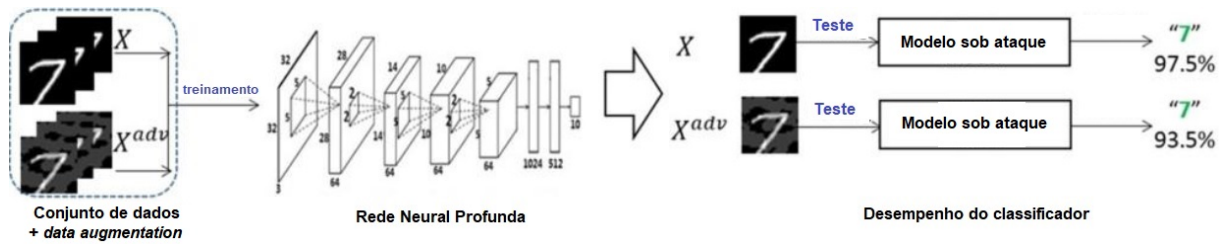


FIG. 13: O *Adversarial Training* aumenta a robustez dos classificadores ao utilizar a técnica de *data augmentation*. Adaptado de Shen et al. (2017).

O *Adversarial Training* utiliza uma variação de *data augmentation*. Em AP, a *data augmentation* é um procedimento que enriquece o conjunto de treinamento ao realizar várias cópias de uma mesma imagem, que posteriormente são modificadas aplicando-se rotações, embaçamento, translações, cortes, alteração na luminosidade, entre outros. Já no *Adversarial Training*, o termo *data augmentation* se refere ao procedimento de inserir imagens contraditórias em um conjunto de treinamento, e foi utilizada por vários trabalhos para aumentar a robustez dos modelos classificatórios (SZEGEDY et al., 2013; GOODFELLOW et al., 2015; HUANG et al., 2015; KURAKIN et al., 2016b; TRAMÈR et al., 2017; ZANTEDESCHI et al., 2017a; MADRY et al., 2017; KANNAN et al., 2018).

Embora o *Adversarial Training* consiga inicialmente produzir bons resultados, este método de defesa possui dois problemas. Um deles consiste que o treinamento com imagens contraditórias geradas a partir de um algoritmo de ataque não produz um modelo genérico, capaz de ser robusto às imagens contraditórias geradas por outro algoritmo de ataque. Para uma maior generalização, seria necessário elaborar um conjunto de treinamento T contendo uma grande quantidade de imagens contraditórias geradas por diferentes algoritmos de ataque, com diferentes quantidades de perturbação. Com isso, surge o segundo problema: o *Adversarial Training* se torna um procedimento computacionalmente custoso, pois além da geração de várias imagens contraditórias a partir de diferentes

¹⁴Para fins didáticos, considere um conjunto de treinamento T formado apenas por uma imagem.

algoritmos de ataque (o que não é garantia de um modelo robusto a uma outra imagem contraditória gerada por um futuro algoritmo de geração mais eficiente), é preciso treinar o modelo em um conjunto de dados várias vezes maior, elevando exponencialmente o tempo de treinamento. Um método de defesa eficiente deve estar desacoplado de qualquer algoritmo de ataque para aumentar sua capacidade de generalização.

Já a Destilação Defensiva é um método de defesa proativo proposto por Papernot et al. (2016c). Este método de defesa é inspirado em uma técnica para a transferência de conhecimento entre modelos de aprendizado conhecida como **destilação** (HINTON et al., 2015). Na destilação de modelos de aprendizado, o conhecimento de um modelo mais complexo, após ser treinado em um determinado conjunto de dados, é transferido para um modelo de aprendizado mais simples. De maneira semelhante, a Destilação Defensiva primeiramente treina um modelo F em um conjunto de amostras X e rótulos Y com uma temperatura t , gerando um vetor de saídas probabilísticas $F(X)$. O conjunto de rótulos Y então é substituído pelo conjunto $F(X)$ e um modelo F^d com a mesma arquitetura de F é criado e treinado com o mesmo conjunto de amostras X e temperatura t , porém com os rótulos probabilísticos $F(X)$. Ao final do treinamento, são produzidas as saídas probabilísticas destiladas $F^d(X)$. A Figura 14 apresenta o modelo esquemático da Destilação Defensiva. Apesar de obter bons resultados contra os ataques JSMA e FGSM, Carlini e Wagner (2017c) mostraram que é possível evadir classificadores com Destilação Defensiva.

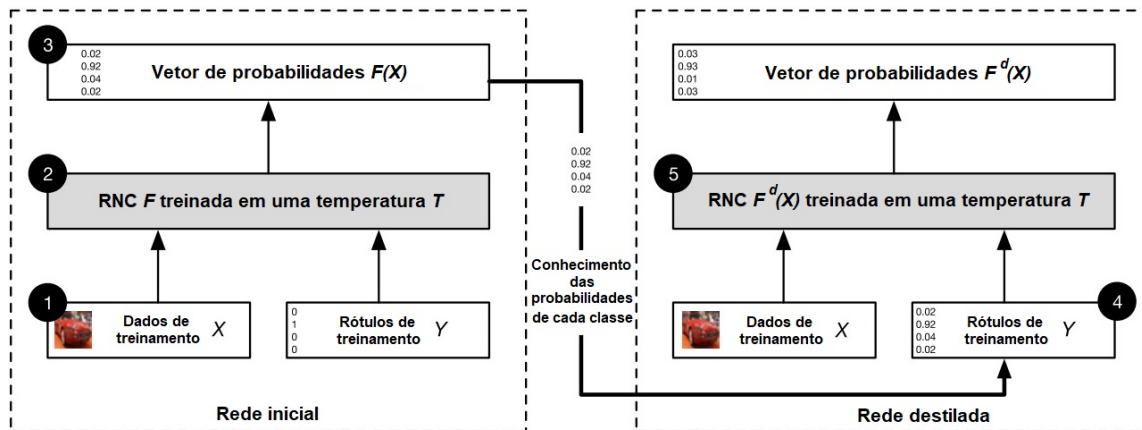


FIG. 14: Modelo esquemático da Destilação Defensiva. Adaptado de (PAPERNOT et al., 2016c, pág. 588).

Métodos de defesa proativos baseados em *Gradient Masking*, como o *Adversarial Training* e a Destilação Defensiva, geralmente produzem um modelo com gradientes mais suaves em certas regiões do espaço, o que torna mais difícil para o atacante encontrar boas direções no espaço para perturbar uma imagem. Entretanto, o atacante pode realizar um

ataque de caixa-preta, ao treinar um modelo substituto. Esse modelo substituto reproduz o comportamento do modelo almejado, uma vez que o atacante monitora as saídas que o modelo almejado atribui para algumas imagens cuidadosamente geradas por ele. O atacante então tira vantagem da propriedade de transferibilidade, e utiliza os gradientes do modelo substituto para gerar imagens contraditórias, que ao serem enviadas ao modelo almejado, também produzam erros de classificação (GOODFELLOW; PAPERNOT, 2017).

3.3 MÉTODOS BASEADOS EM COMITÊS DE CLASSIFICADORES

Métodos de defesa baseados em comitês de classificadores são métodos de defesa proativos formados por dois ou mais modelos de classificação, que podem ou não ser escolhidos em tempo de execução. Essa abordagem se baseia na hipótese de que cada modelo compensa as fragilidades que o outro modelo possui ao classificar uma determinada imagem de entrada (HE et al., 2017). Trabalhos como Strauss et al. (2017); Tramèr et al. (2017); Abbasi e Gagné (2017) e Sengupta et al. (2017) utilizaram diferentes abordagens para a elaboração de métodos de defesa proativos baseados na formação de comitês de classificadores.

Em Sengupta et al. (2017), foi utilizado um algoritmo bayesiano que, a partir das interações entre o defensor e o atacante, faz a escolha de um modelo que minimize as chances de evasão e, ao mesmo tempo, maximize as previsões nas imagens legítimas. Em Strauss et al. (2017) e Abbasi e Gagné (2017) foram formados comitês de RNCs que classificam uma imagem por voto majoritário. Tramèr et al. (2017), por sua vez, utiliza uma variação do *Adversarial Training* para treinar um classificador principal com imagens contraditórias geradas por cada modelo de um comitê de classificadores.

Apesar dos diferentes métodos de defesa proativos baseados em comitês de classificadores, He et al. (2017) mostraram que esse tipo de abordagem está suscetível a evasões por imagens contraditórias, principalmente porque não é adaptativa aos diferentes níveis de conhecimento que o atacante possa ter sobre o seus respectivos *modi operandi*.

3.4 MODELOS AUXILIARES DETECTORES

Trabalhos como Grosse et al. (2017); Gong et al. (2017); Metzen et al. (2017a) e Chen et al. (2017a) elaboraram métodos de defesa reativos baseados em modelos auxiliares detectores (MADs). De forma resumida, métodos de defesa baseados em MADs são métodos de defesa reativos que utilizam o *Adversarial Training* para treinar um modelo classificador

binário em um conjunto de dados contendo imagens legítimas e contraditórias. Após o treinamento, os MADs atuam como um filtro, classificando as imagens de entrada como legítimas ou contraditórias, antes de repassá-las ao classificador de aplicação F .

Grosse et al. (2017) adaptaram o classificador da aplicação F para também desempenhar a função de um MAD, treinando-o em um conjunto de dados contendo $n + 1$ classes. O procedimento consiste em gerar imagens contraditórias x'_i a partir de cada imagem (x_i, y_j) pertencente ao conjunto de treinamento \mathcal{T} , onde $i \leq |\mathcal{T}| \times m$ (sendo m o número de algoritmos de ataque utilizados) e $j \leq n$. Após a geração das imagens contraditórias, foi formado um novo conjunto de treinamento \mathcal{T}_1 , sendo $\mathcal{T}_1 = \mathcal{T} \cup \{(x'_i, n+1), i \leq |\mathcal{T}| \times m\}$. $n + 1$ é o rótulo atribuído a uma imagem contraditória. Por fim, o modelo F foi treinado no conjunto \mathcal{T}_1 .

Já Gong et al. (2017) adotaram um procedimento semelhante ao de Grosse et al. (2017), mas ao invés de adaptarem o classificador da aplicação para classificar as imagens contraditórias em uma nova classe, os autores construíram e treinaram um MAD para filtrar as imagens legítimas X das imagens contraditórias X' (geradas pelos algoritmos de ataque FGSM e JSMA, *i.e.* $m = 2$), utilizando um conjunto de treinamento \mathcal{T}_1 , formado a partir de \mathcal{T} . Formalmente, $\mathcal{T}_1 = \{(x_i, 1) : i \in |\mathcal{T}|\} \cup \{(x'_i, 0) : i \leq |\mathcal{T}| \times m\}$.

Em Metzen et al. (2017a) foram utilizadas as camadas internas de uma RNC para treinar MADs, de uma forma semelhante à abordagem utilizada por Gong et al. (2017). Os autores nomearam esses MADs de **subdetectores** e os fixaram entre diferentes camadas internas da RNC para realizarem a detecção de imagens contraditórias. Neste trabalho foram realizados experimentos utilizando os algoritmos ataque FGSM, BIM e *Deepfool*.

Por fim, Chen et al. (2017a) elaboraram uma arquitetura de detecção e reforma chamada *ReabsNet*. Ao receber uma imagem de entrada x , o *ReabsNet* utiliza um MAD (representado por uma RNC treinada por *adversarial training*) para classificar se x é legítima ou contraditória. Caso seja classificada como legítima pelo MAD, o *ReabsNet* envia a imagem x para o classificador da aplicação. Entretanto, caso seja classificada como contraditória, a imagem x é enviada pelo *ReabsNet* para um processo iterativo, que reforma a imagem x enquanto ela for classificada como contraditória pelo MAD. Ao final do processo de reforma, a imagem x é finalmente enviada para o classificador da aplicação.

Métodos de defesa baseados em MADs são métodos reativos que, basicamente, utilizam o *Adversarial Training* para elaborarem modelos de classificação binários, que atuam como um filtro ao detectarem imagens contraditórias antes de chegarem ao classificador principal. Os MADs podem ser facilmente evadidos, uma vez que (i) são abordagens deterministicas de defesa e, por se basearem no *Adversarial Training*, (ii) ficam fortemente

acoplados aos algoritmos de ataque que foram utilizados no processo de *data augmentation*, realizado nos seus respectivos conjuntos de treinamento.

3.5 MÉTODOS BASEADOS EM ABORDAGENS ESTATÍSTICAS

Algumas defesas como Grosse et al. (2017) e Feinman et al. (2017) realizaram comparações estatísticas entre as distribuições das imagens legítimas e contraditórias. Grosse et al. (2017) elaboraram um método de defesa reativo que utilizou uma aproximação do teste de hipótese MMD (*Maximum Mean Discrepancy*) com o Teste de Permutação de *Fisher* para determinar se um conjunto de imagens legítimas \mathcal{S}_1 pertence à mesma distribuição de um conjunto de dados \mathcal{S}_2 , que pode conter imagens contraditórias. Formalmente, dados dois conjuntos \mathcal{S}_1 e \mathcal{S}_2 , é aplicado inicialmente $a = \text{MMD}(\mathcal{S}_1, \mathcal{S}_2)$. Após, há uma permutação dos elementos de \mathcal{S}_1 e \mathcal{S}_2 em dois novos conjuntos \mathcal{S}'_1 e \mathcal{S}'_2 e defini-se $b = \text{MMD}(\mathcal{S}'_1, \mathcal{S}'_2)$. Se $a < b$, a hipótese nula é rejeitada e é concluído que os dois conjuntos pertencem a distribuições diferentes. Esse processo é repetido várias vezes e o *p-value* é definido como a fração do número de vezes que a hipótese nula foi rejeitada.

Já Feinman et al. (2017) elaborou um método de defesa reativo chamado *Kernel Density Estimation* (KDE). O KDE utiliza um Modelo de Mistura Gaussiano¹⁵ para modelar as saídas da penúltima camada Z de uma RNC (*logits*), e analisar se as imagens contraditórias pertencem a uma distribuição diferente da distribuição das imagens legítimas. De maneira mais formal, dada uma imagem x classificada com o rótulo y , o KDE estima a probabilidade de x como:

$$\text{KDE}(x) = \frac{1}{|X_y|} \sum_{s \in X_y} \exp \left(-\frac{|F^{n-1}(x) - F^{n-1}(s)|^2}{\sigma^2} \right) \quad (9)$$

onde X_y é o conjunto de treinamento com imagens pertencentes à classe y e $F^{n-1}(x)$ é a saída da camada Z referente à imagem de entrada x . O detector é portanto construído através da seleção de um limiar τ , classificando x como contraditório se $\text{KDE}(x) < \tau$ e legítimo caso contrário.

Métodos de defesa reativos baseados em abordagens estatísticas, como os trabalhos de Feinman et al. (2017) e Grosse et al. (2017), também foram evadidos por imagens contraditórias (CARLINI; WAGNER, 2017a). Os dois principais motivos que tornam os métodos baseados em abordagens estatísticas suscetíveis a evasões são (i) suas arquite-

¹⁵Modelos de Mistura Gaussianos (*Gaussian Mixture Models*), são modelos de aprendizado não-supervisionado que representam subpopulações com distribuição normal contidas no interior de uma população geral.

turas determinísticas, e (ii) a similaridade que a distribuição das imagens contraditórias (imagens essas geradas a partir de algoritmos de ataque iterativos mais elaborados, como o CW) geralmente tem com a distribuição das imagens legítimas.

3.6 MÉTODOS BASEADOS EM PRÉ-PROCESSAMENTO

Outros trabalhos elaboraram métodos de defesa baseados em abordagens de pré-processamento de imagens, como o redimensionamento e preenchimento (XIE et al., 2017), Redes Geradoras Contraditórias (SHEN et al., 2017), camadas de ruído (LIU et al., 2017b), *Denoising Autoencoders* (GU; RIGAZIO, 2014) e redução de dimensionalidade (HENDRYCKS; GIMPEL, 2017), (LI; LI, 2016), (XU et al., 2018) e (CARRARA et al., 2018). A seguir, as abordagens adotadas por cada um desses trabalhos são explicadas com mais detalhes.

Em Xie et al. (2017) foi elaborado um método de defesa proativo que insere camadas de redimensionamento e preenchimento no início de uma arquitetura de RNC. A camada de redimensionamento altera as dimensões da imagem de entrada, e após, a camada de preenchimento insere valores nulos em posições aleatórias no entorno da imagem redimensionada. Ao final do processo de preenchimento, a imagem redimensionada e preenchida é classificada pelo modelo proativo. Shen et al. (2017) elaborou um método de defesa proativo que adaptou uma Rede Geradora Contraditória (do inglês, *Generative Adversarial Network* - GAN (GOODFELLOW et al., 2014)) para pré-processar as imagens de entrada antes de serem enviadas aos modelos de classificação. Entretanto, Carlini e Wagner (2017b) mostraram que é possível evadir esta defesa gerando perturbações com o ataque CW. Já Liu et al. (2017b) adotaram uma abordagem formada por camadas de ruído, que são inseridas entre as camadas internas de uma RNC. Essas camadas de ruído aplicam no vetor de entrada correspondente à imagem, um ruído gaussiano gerado de forma aleatória, que segundo os autores, previne ataques baseados em gradiente. Em Gu e Rigazio (2014) foram elaboradas as Redes Contrativas Profundas (do inglês, *Deep Contractive Networks* - DCNs). As DCNs são métodos de defesa proativos que utilizam *denoising autoencoders* como tentativa de remover as perturbações das imagens contraditórias. Apesar das DCNs terem apresentado resultados razoáveis com imagens contraditórias geradas pelo algoritmo L-BFGS, vários outros algoritmos de ataque mais poderosos surgiram desde então, sem que houvessem novos experimentos com as DCNs.

Já há métodos de defesa que pré-processam as imagens de entrada utilizando técnicas de redução de dimensionalidade (HENDRYCKS; GIMPEL, 2017; LI; LI, 2016; XU et al., 2018). Esses trabalhos se baseiam na hipótese de que ao reduzir as dimensões de uma

entrada, a probabilidade de um atacante gerar uma perturbação que afete o desempenho de um classificador diminui, uma vez que o algoritmo de ataque irá ter menos informações referentes ao hiperespaço da imagem de entrada (XU et al., 2018). Com isso em mente, Hendrycks e Gimpel (2017) elaboraram um método de defesa reativo baseado no algoritmo PCA¹⁶. Os autores se basearam na hipótese que as imagens contraditórias inferem um maior peso em componentes principais maiores e um menor peso nos componentes principais iniciais. Já Li e Li (2016) aplicaram o PCA nos valores produzidos pelas camadas de convolução de uma RNC, e usaram um classificador em cascata para detectar imagens contraditórias. O classificador em cascata classifica uma imagem x como legítima somente se todos os classificadores C_i classificarem x como legítima, mas rejeita x se algum classificador C_i rejeitar x . Os autores utilizaram o algoritmo de ataque L-BFGS para realizar os experimentos.

Por fim, Xu et al. (2018) introduziram o *Feature Squeezing*. O *Feature Squeezing* é um método de defesa reativo que utiliza duas técnicas para reduzir a dimensionalidade de uma imagem de entrada: a (i) redução dos *bits* de cor e a (ii) suavização espacial. De acordo com os autores, essas técnicas de redução de dimensionalidade foram escolhidas porque se complementam ao tratarem dois tipos de perturbação em imagens contraditórias. A redução dos *bits* de cor elimina pequenas perturbações em vários *pixels*, enquanto que a suavização espacial elimina grandes perturbações em alguns *pixels*. Para detectar imagens contraditórias, o *Feature Squeezing* gera duas versões reduzidas de uma imagem de entrada x : (i) \hat{x}_1 , que representa a imagem x com os *bits* de cor reduzidos e (ii) \hat{x}_2 , que representa a imagem x reduzida com a suavização espacial. Após, o *Feature Squeezing* envia as imagens x , \hat{x}_1 e \hat{x}_2 para serem classificadas por uma RNC F e compara as saídas *softmax* $F(x)$, $F(\hat{x}_1)$ e $F(\hat{x}_2)$ com o auxílio da métrica de distância L_1 . Se em algum par vetorial, a distância L_1 máxima exceder um limiar pré-definido τ , o *Feature Squeezing* classifica a entrada x como contraditória e a descarta. A Figura 15 apresenta o modelo esquemático do método de defesa. Apesar do *Feature Squeezing* obter resultados promissores na detecção de imagens contraditórias geradas pelos ataques FGSM, JSMA e as variações L_0 , L_2 e L_∞ do CW, He et al. (2017) demonstraram que, devido ao comportamento determinístico do *Feature Squeezing*, é possível gerar uma imagem contraditória que continue contraditória mesmo após a aplicação das técnicas de redução de dimensionalidade de redução de cor e suavização espacial.

No geral, os métodos de defesa baseados em pré-processamento que foram apresen-

¹⁶Análise de Componentes Principais (*Principal Component Analysis* - (PCA)) é uma técnica para redução de dimensionalidade que, ao aplicar uma transformação linear, reduz um conjunto de pontos em um espaço n -dimensional em um novo conjunto de pontos em um espaço k -dimensional, onde $k \leq n$.

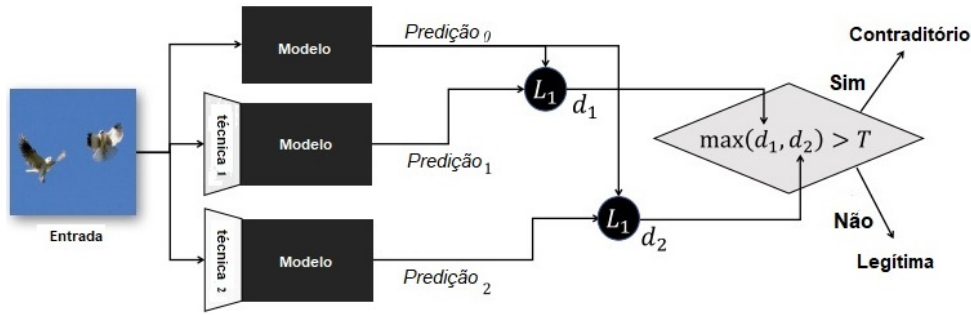


FIG. 15: O funcionamento do *Feature Squeezing* (XU et al., 2018).

tados nesta seção também têm como principal brecha suas arquiteturas determinísticas, que não se adaptam aos diferentes cenários de ataque. Entretanto, eles podem ser mais eficazes ao serem utilizados como parte de uma linha de defesa de uma arquitetura não determinística, por exemplo. Ao atuarem em conjunto, os métodos de defesa baseados em pré-processamento podem auxiliar no processo de detecção, ao tentarem remover algumas perturbações presentes nas imagens contraditórias.

3.7 MÉTODOS BASEADOS EM PROXIMIDADE

Há outros trabalhos como Papernot e McDaniel (2018), Cao e Gong (2017), Carrara et al. (2018) e Meng e Chen (2017) que elaboraram métodos de defesa baseados em medidas de proximidade entre imagens legítimas e contraditórias com a fronteira de decisão. Em Papernot e McDaniel (2018) foi elaborado um método de defesa proativo, chamado *Deep k-Nearest Neighbors* (DkNN). O DkNN utiliza algoritmos de kNN¹⁷ para calcular métricas de incerteza e confiabilidade a partir da proximidade entre as representações internas das imagens de treinamento e as representações internas de uma imagem de entrada x , obtidas por cada camada de uma RNC. Os rótulos que representam os pontos no espaço das imagens de treinamento são analisados após a imagem de entrada percorrer toda a arquitetura da RNC. Caso a predição da imagem de entrada x esteja em conformidade com os rótulos referentes às imagens de treinamento mais próximas a x , a medida de incerteza é baixa. Em contrapartida, caso os rótulos das imagens de treinamento mais próximas estejam divergentes, o valor da incerteza é alto (PAPERNOT, 2018b, pág. 13). A Figura 16 ilustra o funcionamento do DkNN.

Cao e Gong (2017) também utilizaram uma abordagem baseada em medidas de proximidade para elaborar um método de defesa proativo, chamado de *Region-based Clas-*

¹⁷O kNN é um algoritmo de classificação que utiliza uma métrica de distância para calcular quais são os k vizinhos mais próximos de uma entrada x . O kNN então classifica x como sendo pertencente a uma classe c , sendo c a classe mais frequente entre os k vizinhos mais próximos de x .

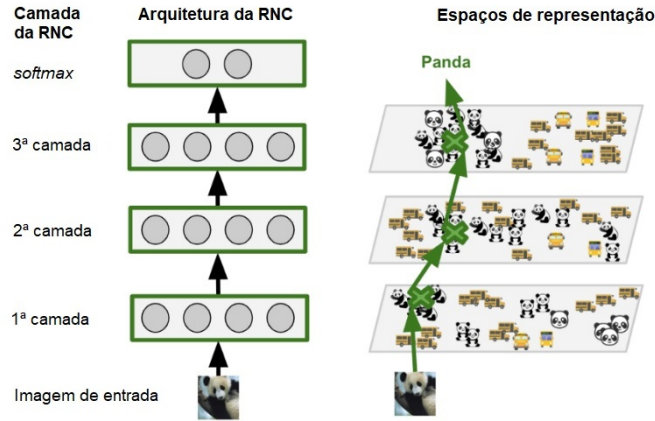


FIG. 16: As DkNNs calculam medidas de incerteza e confiabilidade para suportar uma predição feita por uma RNC para uma determinada imagem de entrada x , ao realizar uma busca entre as imagens de treinamento com representações internas mais próximas às de x . Adaptado de Papernot (2018b, pág. 12).

sification (RC). O RC é uma variação do algoritmo de kNN que define uma região no hiperespaço, tendo como centroide a imagem de entrada x , e a atribui um rótulo correspondente à classe que intersecta a maior área da região do hiperespaço. Formalmente, para uma imagem de entrada x e uma RNC F que divide o hiperespaço em C regiões $R = \{R_1, R_2, \dots, R_C\}$ (sendo C o número de classes e R_i a classe predita para $F(x)$), é criado um hipercubo $B(x, r)$ em torno de x (sendo x o centroide de $B(x, r)$) com comprimento r . $A_i(B(x, r))$ é a área do hipercubo $B(x, r)$ que intersecta a região R_i . O classificador RC formado a partir do modelo F é definido como $RC_{F,r}$ e a sua predição para x é baseada na região R_i que possui a maior interseção com a área do hipercubo, a saber, $RC_{F,r} = \arg \max_i A_i(B(x, r))$. A Figura 17 ilustra graficamente o funcionamento de uma RC.

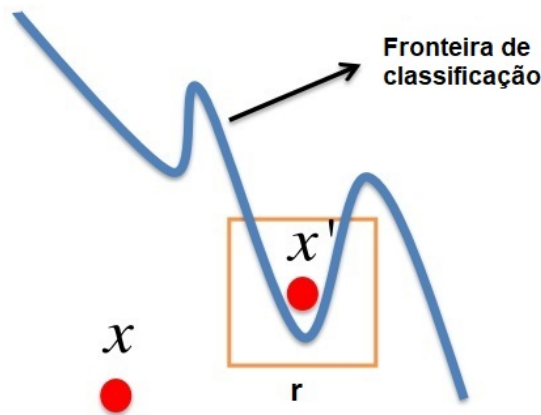


FIG. 17: Ilustração de uma RC. x é uma imagem legítima e x' é sua imagem contraditória correspondente. O hipercubo com centroide em x' intersecta em maior quantidade com a região que possui a imagem legítima x . Adaptado de Cao e Gong (2017, pág. 282).

Já Carrara et al. (2018) elaboraram um método de defesa reativo bem semelhante ao método proativo proposto por Papernot e McDaniel (2018): o DkNN. O método reativo de Carrara et al. (2018) primeiramente utiliza uma RNC F para classificar uma imagem de entrada x . Com isso, as representações internas referentes à imagem x , obtidas de uma camada interna do modelo F (tal camada é escolhida empiricamente), são utilizadas por um algoritmo de kNN para efetuar uma busca no conjunto de treinamento com o objetivo de recuperar as k imagens com representações internas mais semelhantes às representações da imagem x . Com isso, é obtida uma medida de confiança c para a predição $F(x)$, onde c é calculado com base no *score* do algoritmo kNN. Caso a medida de confiança c esteja abaixo de um limiar pré-definido, a imagem x é classificada como contraditória e descartada logo em seguir; caso contrário, a predição $F(x)$ é válida com um nível de confiança c . Apesar desse método de defesa ter conseguido resultados promissores na detecção de imagens contraditórias geradas pelos algoritmos de ataque FGSM e L-BFGS, nenhum experimento foi realizado com imagens contraditórias geradas pelo algoritmo de ataque CW até o momento desta dissertação.

Por sua vez, Meng e Chen (2017) propuseram o *MagNet*: uma arquitetura de defesa reativa não determinística formada por duas camadas de defesa: (i) a camada de detecção, que rejeita as imagens contraditórias que contêm maiores perturbações e, por esse motivo, encontram-se mais distantes da fronteira de decisão, e (ii) a camada de reforma, que reconstrói as imagens oriundas da camada de detecção como tentativa de remover possíveis perturbações que ainda estejam presentes nelas. De acordo com os autores, a camada de reforma atua como um imã, que atrai as imagens contraditórias que evadiram a camada de detecção para a região da fronteira de decisão correspondente à sua verdadeira classe. O *MagNet* escolhe aleatoriamente dois componentes de defesa (implementados como *autoencoders*) a partir de um repositório: um *autoencoder* para a camada de detecção e outro *autoencoder* para a camada de reforma. Ainda de acordo com os autores, a escolha não determinística dos componentes de defesa é inspirada em técnicas de criptografia, como uma forma de diminuir as chances do atacante em evadir a defesa (MENG; CHEN, 2017, pág. 136). Todos os *autoencoders* são treinados com imagens legítimas, o que desacopla a defesa de qualquer algoritmo de ataque. A Figura 18 ilustra o modelo esquemático do *MagNet*.

Formalmente, a camada de detecção recebe uma imagem x e escolhe um *autoencoder* de maneira aleatória ae_d para fazer a reconstrução da imagem $\hat{x} = ae_d(x)$. Após a reconstrução da imagem x , é calculada uma medida me , que pode ser o erro de reconstrução $ER_{x,\hat{x}}$ (ver Equação 2) ou a divergência de *Jensen-Shannon* $JSD_{x,\hat{x}}$ (ver Equação

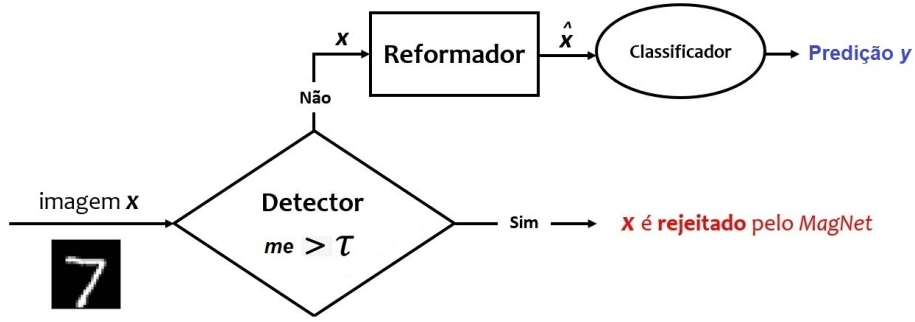


FIG. 18: O modelo esquemático do *MagNet*. Adaptado de Meng e Chen (2017).

8). Formalmente, $me \in \{RE, JSD\}$. Se x for uma imagem legítima, a medida me será pequena. Entretanto, caso x seja uma imagem contraditória, a medida me será maior. Se $me > \tau$, sendo τ um limiar pré-definido a partir de imagens legítimas, a camada de detecção rejeita x . Caso $me \leq \tau$, o *MagNet* passa x para a sua camada de reforma. A camada de reforma recebe a imagem x , oriunda da camada de detecção, e escolhe um outro *autoencoder* ae_r , de maneira aleatória, para realizar a reconstrução da imagem x , como tentativa para a remoção de pequenas perturbações que possam estar inseridas em x . Basicamente, o reformador ideal (i) não deve alterar os resultados da classificação do modelo nas imagens legítimas e (ii) deve reformar as imagens contraditórias de tal maneira que as imagens reconstruídas estejam mais próximas da fronteira de decisão das imagens legítimas. Após o procedimento de reconstrução, a imagem \hat{x} , reformada pelo *autoencoder* ae_r , é enviada ao classificador da aplicação.

Apesar do *MagNet* ter apresentado bons resultados contra os ataques FGSM, BIM, *Deepfool* e CW, Carlini e Wagner (2017b) mostraram que é possível evadir esse método de defesa com perturbações geradas pelo mesmo algoritmo CW em um cenário de ataque de caixa-cinza. As perturbações geradas em Carlini e Wagner (2017b) não foram detectadas pela camada de detecção do *MagNet*, e enganaram o modelo da aplicação mesmo com a reforma das imagens contraditórias. Entretanto, a abordagem não determinística do *MagNet* forçou o atacante a gerar imagens contraditórias com distorções maiores e com um custo computacional elevado. De acordo com Carlini e Wagner, (2017a, pág. 12), defesas baseadas em aleatoriedade podem aumentar significativamente o tamanho das perturbações e o tempo computacional necessário para gerá-las, o que torna essa abordagem uma boa alternativa de defesa contra imagens contraditórias. O mais intrigante é que existem poucos métodos de defesa não determinísticos, o que faz com que essa abordagem seja ainda mais promissora (VOROBAYCHIK; KANTARCIOGLU, 2018, pág. 150), (CARLINI; WAGNER, 2017a, pág. 12). Já ao contrário do *MagNet*, que utilizou *autoencoders* em seus processos de defesa, os outros dois métodos de defesa baseados em proximidade,

que foram propostos por Carrara et al. (2018) e Papernot e McDaniel (2018), utilizaram uma RNC para verificarem se a predição de uma determinada imagem de entrada, feita pela camada *softmax*, está próxima das predições realizadas por cada camada interna. Apesar de não terem sido encontrados trabalhos que simularam ataques nos métodos de defesa de Carrara et al. (2018) e Papernot e McDaniel (2018), acredita-se que por serem determinísticos, seja possível a um atacante (que esteja ciente da presença de algum desses métodos de defesa) gerar imagens contraditórias que consigam evadir a proteção e, com isso, causarem erros de classificação na RNC almejada.

4 O MULTIMAGNET

4.1 VISÃO GERAL

O método elaborado nesta dissertação é chamado *MultiMagNet*. O *MultiMagNet* é inspirado na abordagem não determinística, baseada em proximidade, do *MagNet* para defender classificadores contra imagens contraditórias. O *MultiMagNet* basicamente possui duas etapas: (i) a Etapa de Calibração e (ii) a Etapa de Produção. Durante a Etapa de Calibração, o usuário fornece várias combinações de hiperparâmetros para serem testadas iterativamente em um conjunto de validação. A melhor combinação de hiperparâmetros e os respectivos limiares de classificação são definidos ao final do processo iterativo da Etapa de Calibração e salvos em um arquivo de configuração.

Já na Etapa de Produção, o *MultiMagNet* carrega do arquivo de configuração, a melhor combinação de hiperparâmetros e, em seguida, forma aleatoriamente comitês de *autoencoders* que pré-processam e classificam por voto majoritário, com o auxílio dos limiares definidos pela Etapa de Calibração, as imagens de entrada como legítimas ou contraditórias. A formação aleatória de um comitê de *autoencoders* visa aumentar a diversidade e o efeito não determinístico do *MultiMagNet* em comparação ao *MagNet*. Após, de forma semelhante ao *MagNet*, as imagens de entrada que passaram pela camada de detecção são enviadas para a camada de reforma onde são reconstruídas por um outro *autoencoder*, também escolhido de forma aleatória. Finalmente, as versões reconstruídas pela camada de reforma são enviadas para o classificador da aplicação. A seguir, cada processo que forma as Etapas de Calibração e Produção são explicados formalmente.

4.2 ETAPA DE CALIBRAÇÃO

Durante a Etapa de Calibração, todas as combinações de hiperparâmetros, predefinidas pelo usuário em um arquivo de entrada C , são avaliados com a ajuda do (i) conjunto de validação V , (ii) do classificador da aplicação F e (iii) do repositório S contendo m *autoencoders*. No final da Etapa de Calibração, a melhor combinação de hiperparâmetros $c_b \in C$ e o conjunto dos limiares $T_b = \{\tau_{1b}, \tau_{2b}, \dots, \tau_{mb}\}$, oriundos do cálculo de cada um dos m *autoencoders* do repositório S com a melhor combinação de hiperparâmetros c_b , são então salvos em um arquivo chamado C_{best} . Ao ser definido, o arquivo C_{best} fornecerá prontamente à Etapa de Produção a melhor combinação de hiperparâmetros c_b

e o conjunto dos limiares T_b , para que sejam evitados cálculos desnecessários e repetitivos. Todos os processos relacionados à Etapa de Calibração, que estão ilustrados pela Figura 19, serão formalizados a seguir.

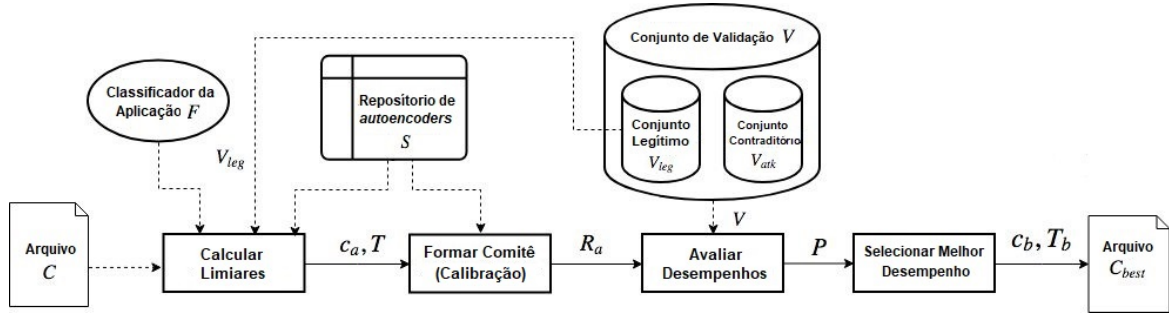


FIG. 19: Modelo esquemático da Etapa de Calibração.

4.2.1 CALCULAR LIMIARES

O primeiro processo da Etapa de Calibração, Calcular Limiares, recebe como entrada um arquivo C que contém todos os conjuntos de valores, predefinidos empiricamente pelo usuário, para cada um dos seguintes hiperparâmetros do *MultiMagNet*:

- (i) **a taxa de falsos positivos**, cujos valores são representados pelo conjunto T_{fp} , tal que $t_{fp_a} \in T_{fp}$ e $0 < t_{fp_a} < 1$;
- (ii) **a métrica para o cálculo dos limiares**, representado pelo conjunto M , tal que $me_a \in M$ e $me_a \in \{RE, JSD\}$. Quando $me_a = RE$ os limiares são calculados utilizando a métrica baseada no erro de reconstrução definida pela Equação 2. Já quando $me_a = JSD$, é utilizada a métrica baseada na divergência de *Jensen-Shannon*, definida pela Equação 8;
- (iii) **o valor da temperatura**, representados pelos valores do conjunto K , onde $k_a \in K, k_a \geq 1$. Caso $me_a = RE, k = 1$;
- (iv) **a abordagem do limiar**, que é representada pelo conjunto \mathcal{T} , onde $\delta_a \in \mathcal{T}$, $\mathcal{T} = \{minTA, MTA\}$.

A finalidade de cada hiperparâmetro é explicada no decorrer desta seção e da Seção 4.2.3. Além do arquivo C , o processo Calcular Limiares também recebe como entradas o conjunto de validação contendo imagens legítimas V_{leg} , o repositório S contendo m *autoencoders*¹⁸ e o classificador da aplicação F . Este processo produz como saída o conjunto

¹⁸Todos os m *autoencoders* no repositório S são previamente treinados no conjunto de treinamento Tr , que contém somente imagens legítimas.

T que contém $m \times c$ limiares, onde m é o número de *autoencoders* no repositório S , c é o número de combinações possíveis dos valores predefinidos no arquivo C para os conjuntos de hiperparâmetros T_{fp} , K , M e \mathcal{T} , tal que $c = |T_{fp}| \times |K| \times |M| \times |\mathcal{T}|$, e $\mathbf{a} \leq \mathbf{c}$ representa o índice do processo iterativo.

Para calcular todos os $m \times c$ limiares do conjunto T , as imagens legítimas pertencentes ao conjunto de validação V_{leg} foram inicialmente reconstruídas para cada *autoencoder* $s_i \in S, i \leq m$, formando, desta forma, o conjunto V_{L_i} , onde $V_{L_i} = \{vl_i | vl_i = s_i(v_l), v_l \in V_{leg}\}$, sendo $s_i(v_l)$ a imagem v_l reconstruída pelo *autoencoder* s_i . Após, os limiares de classificação são calculados a partir da métrica RE ou da métrica JSD , de acordo com o valor atual do hiperparâmetro me_a . Se $me_a = RE$, a Equação 2 é utilizada para calcular os erros de reconstrução entre cada imagem legítima v_l e sua versão reconstruída vl_i , formando, desta forma, o vetor RE_i que, de acordo com a Equação 2, $x = v_l$, $ae(x) = vl_i$ e $p = 1$. Após calcular o erro de reconstrução de cada imagem legítima pertencente ao conjunto V_{leg} utilizando o *autoencoder* s_i , os valores do vetor RE_i são então ordenados em ordem decrescente e o hiperparâmetro $t_{fpa} \in T_{fp}$ é aplicado. A taxa de falsos positivos t_{fpa} representa a porcentagem de imagens legítimas que podem ser classificadas incorretamente como contraditórias. Portanto, ao ser aplicado no vetor RE_i , o hiperparâmetro t_{fpa} define o limiar τ_i , onde $\tau_i = RE_i[t]$, sendo t o índice calculado a partir do hiperparâmetro t_{fpa} , tal que $t = t_{fpa} \times |V_{leg}|$.

Em contrapartida, caso $me_a = JSD$, é necessário obter as saídas probabilísticas *softmax* F da última camada do classificador da aplicação (representado por uma RNC) para cada imagem legítima do conjunto V_{leg} e sua respectiva versão reconstruída. Vale a pena ressaltar que as saídas *softmax* representam as classes pertencentes ao problema de classificação original. A Equação 10 descreve a função *softmax* utilizada.

$$F(Z_i) = \frac{\exp(Z_i/k_a)}{\sum_{j=1}^n \exp(Z_j/k_a)} \quad (10)$$

Na Equação 10, F representa a camada *softmax* do classificador da aplicação, Z representa as saídas *logits* da penúltima camada do classificador da aplicação, i representa o índice correspondente da classe do problema de classificação original e n é o número total de classes do problema de classificação original. O hiperparâmetro $k_a \in K$ (onde $k_a \geq 1$) é usado para normalizar os valores de Z_i e Z_j , para prevenir que haja saturação (MENG; CHEN, 2017). Com as saídas *softmax* do classificador da aplicação correspondentes às imagens legítimas do conjunto V_{leg} e suas versões reconstruídas V_{L_i} , é calculada a divergência de *Jensen-Shannon* (JSD) a partir da Equação 8, onde $P = F(v_l)$, $Q = F(vl_i)$. De acordo com Meng e Chen (2017), a utilização da mé-

trica JSD é baseada na hipótese que a divergência entre uma imagem legítima qualquer x_{leg} e sua versão reconstruída xr_{leg} é geralmente menor que a divergência entre as saídas *softmax* de uma imagem contraditória x_{adv} e sua versão reconstruída xr_{adv} , tal que $JSD(F(x_{leg}), F(xr_{leg})) < JSD(F(x_{adv}), F(xr_{adv}))$. De uma maneira semelhante ao cálculo do limiar para a métrica *RE*, todas as divergências calculadas utilizando o conjunto de validação legítimo V_{leg} e o *autoencoder* s_i são armazenadas em um vetor chamado JSD_i , que também é ordenado em ordem decrescente e $t_{fpa} \in T_{fp}$ é aplicado, produzindo como resultado o limiar $\tau_i = JSD_i[t]$, onde $t = t_{fpa} \times |V_{leg}|$.

Portanto, ao utilizar os hiperparâmetros $k_a \in K$, $me_a \in \{RE, JSD\}$ e $t_{fpa} \in T_{fp}$ (onde $a \leq c$ e $c = |T_{fp}| \times |K| \times |M| \times |\mathcal{T}|$), todos predefinidos em um arquivo C , é produzido como saída o conjunto T , contendo $m \times c$ limiares. O hiperparâmetro $\delta_a \in \mathcal{T}$, que representa a abordagem do limiar, é utilizado somente pelo processo Avaliar Desempenhos (veja a Seção 4.2.3).

4.2.2 FORMAR COMITÊ (CALIBRAÇÃO)

O processo Formar Comitê (Calibração) recebe como entradas (i) o repositório S contendo m diferentes *autoencoders*, (ii) o conjunto T contendo $m \times c$ limiares e (iii) a combinação atual c_a a ser testada no conjunto de validação V , onde $c_a \in C$ e $c_a = (t_{fpa}, k_a, me_a, \delta_a)$, $a \leq c$. Este processo produz como saída um comitê R_a , que é formado como descrito a seguir: n *autoencoders* são escolhidos aleatoriamente do repositório S , onde $n < m$ e $n \bmod 2 = 1$ (para garantir que não hajam empates na contagem dos votos). Após escolher os n *autoencoders*, seus respectivos limiares também são selecionados a partir do conjunto T , de acordo com a combinação atual c_a , formando, desta maneira, o conjunto R_a . Formalmente, $R_a = \{r_{1a}, r_{2a}, \dots, r_{na}\}$, onde r_{ia} é o par (r_i, τ_{ia}) , tal que $r_i \in S$, $i \leq n$ e $\tau_{ia} \in T$, $a \leq c$.

4.2.3 AVALIAR DESEMPENHOS

O processo Avaliar Desempenhos recebe duas entradas: (i) o comitê R_a e (ii) o conjunto de validação $V = V_{leg} \cup V_{atk}$. V_{atk} é formado por 2.000 imagens contraditórias geradas a partir das 2.000 imagens legítimas em V_{leg} , utilizando um algoritmo de ataque *atk*, onde $atk \in \{FGSM, BIM, DeepFool, CW\}$ (para mais detalhes, consulte a Seção 5.3).

Cada uma das 4.000 imagens em V é reconstruída por cada *autoencoder* do conjunto R_a , formando o conjunto VR_a . Após, é utilizado a Equação 2 ou a Equação 8, de acordo com a métrica me_a atual, para calcular o conjunto de valores M_{V_a} . Se $me_a = RE$, é aplicada a Equação 2, onde $x = v_i$, $ae(x) = vr_i$ e $p = 1$, tal que $v_i \in V$, $vr_i \in VR_a$. Se

$me_a = JSD$, é aplicada a Equação 8, onde $P = F(v_i)$ e $Q = F(vr_i)$, $v_i \in V$, $vr_i \in VR_a$.

Finalmente, cada valor $m_i \in M_{V_a}$ é comparado com o limiar τ_a , que é calculado baseado no valor atual da abordagem do limiar δ_a , que pode ser: (i) a abordagem do limiar mínimo (*minTA*) e (ii) a abordagem do limiar múltiplo (*MTA*). Quando $\delta_a = minTA$, é considerado o menor limiar entre todos os limiares associados aos n *autoencoders* em R_a , *i.e.* $\tau_a = \min\{\tau_{1a}, \tau_{2a}, \dots, \tau_{na}\}$. Em contrapartida, quando $\delta_a = MTA$, $\tau_a \in \{\tau_{1a}, \tau_{2a}, \dots, \tau_{na}\}$. Quando $m_i \leq \tau_a$, uma variável q_{leg} (que representa a quantidade de votos para v_i ser considerada uma imagem legítima), conta um voto, caso contrário, q_{adv} conta um voto. Por maioria de votos, a imagem v_i é finalmente classificada como legítima se $q_{leg} > q_{atk}$, e classificada como contraditória caso $q_{leg} < q_{atk}$. No final da contagem de votos para cada imagem $v_i \in V$, é produzida uma matriz de confusão Ma , de acordo com a Equação 11.

$$Ma = \begin{bmatrix} TN & FN \\ FP & TP \end{bmatrix} \quad (11)$$

Na Equação 11, os elementos TN, FN, FP e TP da matriz Ma definem, respectivamente, o total de (i) imagens contraditórias classificadas como contraditórias, (ii) imagens contraditórias classificadas como legítimas, (iii) imagens legítimas classificadas como contraditórias e (iv) imagens legítimas classificadas como legítimas. Após calcular as c matrizes de confusão, o processo Avaliar Desempenhos finalmente produz como saída o conjunto P , formado por c tuplas, onde cada tupla contém as seguintes três métricas: (i) Acurácia (ACC), (ii) Valor Preditivo Positivo (VPP) e (iii) Valor Preditivo Negativo (VPN). Essas três métricas são formalmente definidas pela Equação 12.

$$\begin{aligned} ACC &= \frac{TN+TP}{TN+FN+FP+TP} \\ VPN &= \frac{TN}{TN+FN} \\ VPP &= \frac{TP}{TP+FP} \end{aligned} \quad (12)$$

É importante destacar que as métricas VPP e VPN foram adotadas para medirem, respectivamente, a capacidade da defesa em classificar as imagens legítimas como legítimas e as imagens contraditórias como contraditórias. Já a acurácia foi adotada com o intuito de fornecer uma visão geral do desempenho da defesa na classificação das imagens.

4.2.4 SELECIONAR MELHOR DESEMPENHO

O processo Selecionar Melhor Desempenho recebe como entrada o conjunto P contendo as c tuplas de desempenho. É produzido como saída o arquivo C_{best} , que contém (i) a

combinação de hiperparâmetros c_b que produziu a maior acurácia no *MultiMagNet*, onde $c_b = (t_{f_{pb}}, k_b, m_{e_b}, \delta_b)$ e (ii) o conjunto T_b , tal que $T_b \subset T$. O conjunto T_b possui os n limiares calculados a partir da combinação c_b de hiperparâmetros, onde $c_b \in C$. A maior acurácia no conjunto de validação V foi escolhida para eleger a melhor combinação de hiperparâmetros c_b , uma vez que essa métrica descreve um melhor cenário de desempenho do *MultiMagNet*.

4.3 ETAPA DE PRODUÇÃO

Uma vez calibrado, o *MultiMagNet* está pronto para receber as imagens de entrada. Já na Etapa de Produção, o *MultiMagNet* primeiramente carrega do arquivo C_{best} a melhor combinação de hiperparâmetros c_b e os seus respectivos limiares T_b , definidos durante a Etapa de Calibração. Ao receber uma imagem de entrada x , o *MultiMagNet* então forma um comitê R contendo n *autoencoders* e utiliza a combinação de hiperparâmetros c_b e os n limiares do conjunto T_b , referentes a cada *autoencoder* do comitê R , para enfim retornar um veredito se a imagem x recebida é, de fato, legítima ou contraditória.

A classificação feita pelo *MultiMagNet* é por voto majoritário. Caso seja classificada como legítima pela maioria dos limiares referentes aos n *autoencoders* do comitê R , a imagem x é reconstruída por um outro *autoencoder*, aleatoriamente escolhido do repositório R ¹⁹. Após a reforma, a imagem x finalmente é enviada para o classificador da aplicação F . Caso a imagem x seja classificada como contraditória, ela é descartada pelo *MultiMagNet* antes de chegar ao classificador da aplicação. A Figura 20 ilustra os cinco processos referentes à Etapa de Produção, e cada um deles será explicado a seguir.

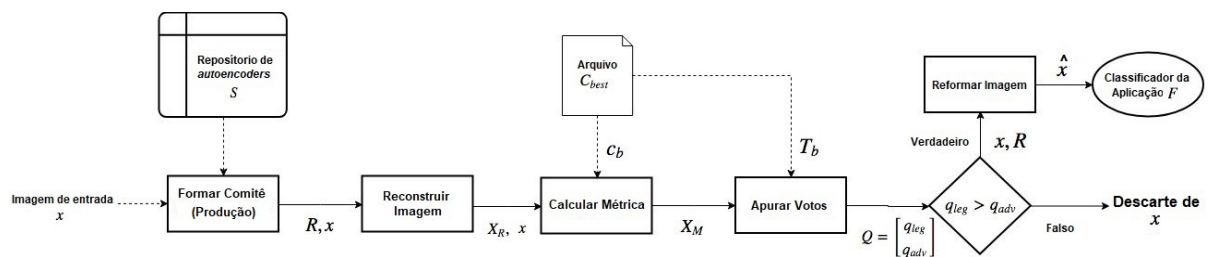


FIG. 20: Modelo esquemático da Etapa de Produção.

¹⁹A reforma da imagem x é uma forma de proteção complementar, que visa remover quaisquer perturbações que possam estar presentes em x e que não foram detectadas pelo comitê R .

4.3.1 FORMAR COMITÊ (PRODUÇÃO)

Esse processo recebe duas entradas: (i) o repositório S contendo m *autoencoders* e (ii) a imagem de entrada x a ser avaliada pelo *MultiMagNet*. Produz como saída um comitê R contendo n *autoencoders* escolhidos aleatoriamente, onde $n \leq m, n \bmod 2 = 1$.

4.3.2 RECONSTRUIR IMAGEM

O processo Reconstruir Imagem recebe duas entradas: (i) a imagem de entrada x e (ii) o comitê R . Produz como saídas a própria imagem x e o conjunto X_R , que é formado pelas versões reconstruídas \hat{x}_i da imagem de entrada x feitas por cada *autoencoder* $r_i \in R$.

4.3.3 CALCULAR MÉTRICA

O processo Calcular Métrica recebe três entradas: (i) a imagem de entrada x , (ii) o conjunto X_R contendo as n versões reconstruídas de x , tal que $\hat{x}_i \in X_R$, e (iii) o arquivo C_{best} , que contém a combinação de hiperparâmetros c_b que levou o *MultiMagNet* a obter o melhor desempenho na Etapa de Calibração. De uma forma bem semelhante aos procedimentos definidos nas Seções 4.2.1 e 4.2.3, o processo Calcular Métrica produz como saída o conjunto X_M , que contém os valores obtidos entre a imagem de entrada x e suas versões reconstruídas $\hat{x}_i \in X_R, i \leq n$. Caso $me_b = RE$, os valores são calculados a partir da Equação 2, onde $ae(x) = \hat{x}_i$ e $p = 1$. Já no caso de $me_b = JSD$, os valores são calculados a partir da Equação 8, onde $P = F(x)$ e $Q = F(\hat{x}_i)$.

4.3.4 APURAR VOTOS

O processo Apurar Votos adota uma abordagem bem semelhante à apuração de votos realizada pelo processo da Etapa de Calibração, Avaliar Desempenho (veja a Seção 4.2.3). O processo Apurar Votos recebe como entradas os conjuntos X_M e T_b , este fornecido pelo arquivo C_{best} , e produz como saída o vetor Q , que contém os valores de q_{leg} e q_{adv} , que respectivamente representam a contagem de votos de x ser uma imagem legítima ou contraditória. Se $q_{leg} < q_{adv}$, a imagem x é descartada e se $q_{leg} > q_{adv}$ é enviada ao último processo: Reformar Imagem.

4.3.5 REFORMAR IMAGEM

O último processo, Reformar Imagem, recebe como entradas: (i) a imagem x classificada como legítima pelo comitê de *autoencoders* R e (ii) o próprio comitê R . Após receber a imagem x , é escolhido aleatoriamente de R um *autoencoder* r_i que reconstrói a imagem x e

produz uma imagem reconstruída \hat{x} , tal que $\hat{x} = r_i(x)$. Finalmente, a imagem reconstruída \hat{x} é enviada para ser classificada pelo classificador da aplicação F .

5 EXPERIMENTOS E RESULTADOS

5.1 VISÃO GERAL

Com o objetivo de validar a hipótese levantada por este trabalho, foi realizado um estudo comparativo entre o *MagNet* e o *MultiMagNet*. Esse estudo buscou avaliar o desempenho de ambos os métodos de defesa na proteção dos classificadores da aplicação contra a ação de imagens contraditórias geradas a partir dos quatro principais algoritmos de ataque, a saber, FGSM, BIM, *DeepFool* e CW. Os experimentos foram conduzidos em um cenário de ataque de caixa-cinza, onde foram formadas quatro variações de um conjunto de teste D , contendo 2.000 imagens legítimas e 2.000 imagens contraditórias. A formação do conjunto de teste D foi realizada para cada *dataset* de imagens utilizado nos experimentos: (i) o MNIST (LECUN et al., 1998) e (ii) o CIFAR-10 (KRIZHEVSKY; HINTON, 2009) (veja a Seção 5.2). O estudo comparativo foi dividido em dois Conjuntos de Experimentos (CEs) principais:

CE I : avaliação do desempenho das respectivas camadas de detecção do *MagNet* e *MultiMagNet* em 40 experimentos no conjunto D ;

CE II : avaliação do classificador da aplicação sem e com a presença do *MagNet* e *MultiMagNet* em 40 experimentos no conjunto D .

O CE I avaliou se o aumento da diversidade e do efeito não determinístico, resultantes da formação aleatória de comitês de *autoencoders* em tempo de execução, proporcionaram ao *MultiMagNet* uma maior capacidade de detecção de imagens contraditórias em relação ao *MagNet*, que escolhe, aleatoriamente, apenas um *autoencoder* em tempo de execução. Em contrapartida, o CE II verificou como o desempenho do classificador da aplicação foi afetado pela influência das imagens contraditórias e o quanto as presenças do *MagNet* e do *MultiMagNet* foram eficazes em protegê-lo. Para avaliar todas as possibilidades, o CE II foi dividido em cinco Cenários: (i) acurácia do classificador da aplicação sem nenhuma defesa presente, (ii) acurácia do classificador da aplicação com a camada de detecção do *MagNet* presente, (iii) acurácia do classificador da aplicação com a camada de detecção do *MultiMagNet* presente, (iv) acurácia do classificador da aplicação com a camada de detecção do *MagNet* e a camada de reforma presentes e, finalmente, (v) acurácia do classificador da aplicação com a camada de detecção do *MultiMagNet* e a

camada de reforma presentes. Em cada Cenário, foram formados, a cada 100 imagens do conjunto D , um novo comitê de *autoencoders*, totalizando 40 experimentos. Para cada comitê formado, foram sorteados n *autoencoders*, sendo n um número aleatório, tal que $1 \leq n < 10$ e $n \bmod 2 = 1$. Os resultados de cada CE, presentes nas Seções 5.5.1 e 5.5.2, apontam para a validade da hipótese levantada por este trabalho.

As próximas seções deste capítulo estão organizadas da seguinte forma: a Seção 5.2 apresenta os *datasets* MNIST e CIFAR-10, que foram utilizados para a realização dos experimentos; a Seção 5.4, além de listar as ferramentas e os equipamentos adotados, traz outras informações sobre a implementação do protótipo do *MultiMagNet*. Por fim, a Seção 5.5 formaliza a metodologia empregada em cada Conjunto de Experimentos, além de apresentar e discutir os resultados obtidos.

5.2 DATASETS

O MNIST (LECUN et al., 1998) é um *dataset* formado por imagens legítimas de dígitos manuscritos correspondentes aos algarismos de 0 a 9. Todas as imagens do MNIST possuem dimensões $28 \times 28 \times 1$, que representam, respectivamente, 28 *pixels* de largura, 28 *pixels* de altura e 1 canal de cor. O MNIST possui um total de 60.000 imagens, que originalmente estão repartidas em 50.000 imagens para treinamento e 10.000 imagens para avaliação. Cada imagem do MNIST é rotulada em uma das 10 diferentes classes do problema original, onde cada classe representa um dígito. A Figura 21 ilustra algumas imagens do MNIST.

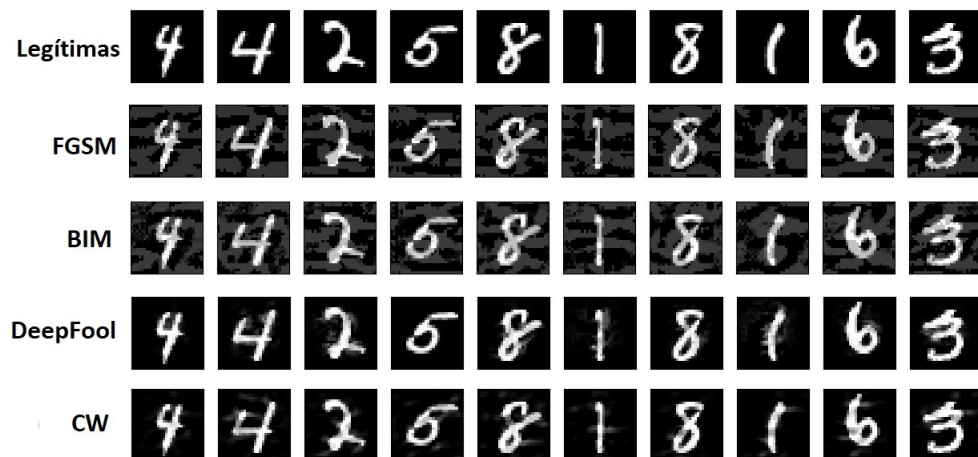


FIG. 21: Exemplos de imagens do MNIST. A primeira linha contém imagens legítimas. As demais apresentam as versões perturbadas dessas imagens pelos ataques FGSM, BIM, *DeepFool* e CW, respectivamente.

O CIFAR-10 também é formado por 60.000 imagens, mas em contrapartida ao MNIST,

as imagens do CIFAR-10 são coloridas e possuem dimensões de $32 \times 32 \times 3$, que respectivamente representam 32 *pixels* de largura, 32 *pixels* de altura e 3 canais de cor. Originalmente, as 60.000 imagens do CIFAR-10 são particionadas em 50.000 imagens para treinamento e 10.000 imagens para avaliação. Cada imagem é rotulada em uma das 10 diferentes classes do problema de classificação original: (i) avião, (ii) automóvel, (iii) ave, (iv) gato, (v) veado, (vi) cachorro, (vii) sapo, (viii) cavalo, (ix) navio e (x) caminhão. Algumas amostras do CIFAR-10 são ilustradas pela Figura 22. Vale ressaltar que os *datasets* MNIST e CIFAR-10 foram adotados na realização dos experimentos uma vez que são amplamente utilizados em vários trabalhos relacionados (XU et al., 2018), (MENG; CHEN, 2017), (ZANTEDESCHI et al., 2017b), (CARLINI; WAGNER, 2017c).

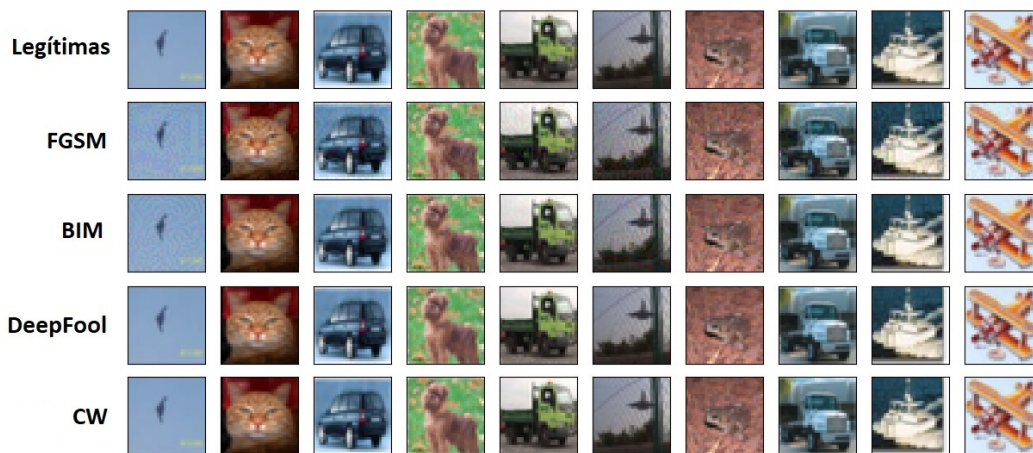


FIG. 22: Exemplos de imagens do CIFAR-10. A primeira linha contém imagens legítimas. As demais apresentam as versões perturbadas dessas imagens pelos ataques FGSM, BIM, *DeepFool* e CW, respectivamente.

É importante mencionar que todas as imagens dos *datasets* MNIST e CIFAR-10 foram normalizadas para terem a intensidade dos seus *pixels* no intervalo $[0, 1]$, ao invés do intervalo original $[0, 255]$. A normalização dos dados nesse intervalo contribuiu para o treinamento dos classificadores e dos *autoencoders*.

5.3 PARTICIONAMENTO DOS DADOS

Nos experimentos realizados em cada Conjunto de Experimentos, ambos os *datasets* foram particionados da seguinte forma: as primeiras 45.000 imagens do conjunto de treinamento original passaram a formar um novo conjunto chamado Tr , que ficou destinado ao treinamento dos m *autoencoders* (pertencentes ao repositório S) e do classificador da aplicação F . Já para a formação do conjunto de validação V , foram escolhidas aleatoriamente 2.000 imagens legítimas das 5.000 imagens restantes do conjunto de treinamento original.

O motivo da escolha de apenas 2.000 imagens está relacionado ao alto custo computacional dos algoritmos de ataque em gerar todas as 5.000 imagens contraditórias. Essas 2.000 imagens escolhidas foram rotuladas como legítimas e armazenadas no conjunto de validação V_{leg} . Após, foram gerados, a partir das imagens de V_{leg} , quatro conjuntos contendo somente imagens contraditórias, que foram denominados V_{FGSM} , V_{BIM} , $V_{DeepFool}$ e V_{CW} . Cada conjunto corresponde às 2.000 imagens rotuladas como contraditórias, geradas respectivamente pelos algoritmos de ataque FGSM, BIM, *DeepFool* e CW. A união dos conjuntos de validação V_{leg} e umas das variações V_{atk} formaram o conjunto de validação V , que foi utilizado durante a Etapa de Calibração (veja a Figura 19). Formalmente, $V = V_{leg} \cup V_{atk}$, onde $atk \in \{FGSM, BIM, DeepFool, CW\}$.

Finalmente, as 10.000 imagens, originalmente separadas para fins de teste, formaram o conjunto Te . O conjunto Te , contendo todas as 10.000 imagens legítimas, foi utilizado para avaliar o classificador da aplicação e os *autoencoders* após o treinamento. Já para avaliar os métodos de defesa *MagNet*²⁰ e *MultiMagNet*, foi necessário definir um subconjunto de teste chamado D , devido ao alto custo computacional em gerar as 10.000 imagens contraditórias. De forma semelhante à formação do conjunto de validação V , foram selecionadas aleatoriamente 2.000 imagens do conjunto Te , que foram rotuladas como legítimas e armazenadas no conjunto D_{leg} . Após, cada algoritmo de ataque gerou 2.000 imagens contraditórias, seguindo o mesmo critério adotado na formação do conjunto de validação V : cada um dos quatro algoritmos de ataque, a saber, FGSM, BIM, *DeepFool* e CW, foi aplicado nas imagens legítimas de D_{leg} , e as imagens resultantes foram armazenadas respectivamente nos conjuntos D_{FGSM} , D_{BIM} , $D_{DeepFool}$ e D_{CW} . Por fim, a união dos conjuntos D_{leg} e D_{atk} formaram o conjunto D , utilizado durante as Etapas de Produção do *MultiMagNet* (veja a Figura 20) e *MagNet*. Formalmente, $D = D_{leg} \cup D_{atk}$, onde $atk \in \{FGSM, BIM, DeepFool, CW\}$.

5.4 PROTÓTIPO

O protótipo do *MultiMagNet* foi desenvolvido na linguagem *Python*, utilizando as bibliotecas *TensorFlow*, *Keras*, *SciKit-Learn*, *NumPy* e *SciPy*. Para a geração das imagens contraditórias, foi utilizado o *framework* ART (*Adversarial Robustness Toolbox*) (NICOLAIE et al., 2018), que contém as implementações dos algoritmos de ataque FGSM, BIM, *DeepFool* e CW. A arquitetura do classificador da aplicação para o *dataset* MNIST obteve, após o treinamento, uma acurácia de 99,40% no problema de classificação original. Já a

²⁰O funcionamento do *MagNet* é simulado nas ocasiões em que o *MultiMagNet* seleciona aleatoriamente apenas um *autoencoder* do repositório S . A seleção de apenas um *autoencoder* é uma característica inerente do *MagNet* e não uma escolha de implementação.

arquitetura do classificador da aplicação adotada para o *dataset* CIFAR-10, a saber, uma *All Convolutional Net* (SPRINGENBERG et al., 2014), apresentou, após o treinamento, uma acurácia de 89,76% no problema de classificação original. Para mais informações sobre as arquiteturas e hiperparâmetros utilizados em ambos os classificadores, consulte as Tabelas 7 e 8. Todos os *autoencoders* do repositório *S* para os *datasets* MNIST (veja a Tabela 9) e CIFAR-10 (veja a Tabela 10) apresentaram, após o treinamento, erros de reconstrução abaixo de 0,001. É importante ressaltar que todo o código-fonte²¹, juntamente com as imagens contraditórias utilizadas nos experimentos²², estão disponíveis *online*. Todos os experimentos foram realizados em uma única máquina com a seguinte configuração: um processador *Core i7 3770*, 16GB de memória RAM e uma GPU GTX 1060 6GB com 1280 *CUDA cores*.

5.5 RESULTADOS

As Seções 5.5.1 e 5.5.2 detalham a metodologia empregada nos Conjuntos de Experimentos I e II, além de apresentarem e discutirem os resultados obtidos.

Para cada algoritmo de ataque, foram utilizados os hiperparâmetros presentes na Tabela 11. Os hiperparâmetros de cada algoritmo de ataque foram empiricamente definidos, de modo que as imagens contraditórias geradas por eles possuíssem pequenas perturbações (veja as Figuras 21 e 22) e, ao mesmo tempo, conseguissem afetar a capacidade de classificação dos modelos da aplicação. A Tabela 2 apresenta as acurácias do classificador da aplicação nos conjuntos T_e (contendo as 10.000 imagens legítimas para teste) e D_{atk} (contendo 2.000 imagens contraditórias), para cada *dataset*. É possível perceber como a capacidade de classificação do modelo foi afetada pela influência das imagens contraditórias, com destaque para os algoritmos de ataque *DeepFool*, aplicado no *dataset* MNIST e CW, aplicado no *dataset* CIFAR-10. Já a Tabela 3 apresenta a melhor combinação de hiperparâmetros c_b , definida pela Etapa de Calibração do *MultiMagNet* para cada algoritmo de ataque. Vale ressaltar que, para os experimentos com o *MagNet*, foram utilizados os mesmos valores presentes na Tabela 3 para os hiperparâmetros **Taxa de Falsos Positivos**, **Métrica** e **Temperatura**. Entretanto, o hiperparâmetro **Abordagem do Limiar** não foi empregado no *MagNet*, uma vez que esse método de defesa escolhe apenas um *autoencoder* em tempo de execução.

²¹<https://github.com/gabrielrmachado/MultiMagNet>

²²<https://drive.google.com/open?id=115KHwpbWLLgcv34AGUF3Fq3z3fuXV22z>

²³Hiperparâmetro indisponível para o *MagNet*.

TAB. 2: Acurácias do classificador da aplicação nos conjuntos T_e e D_{atk} , sem nenhuma defesa presente.

<i>Dataset</i>	Acurácia conjunto T_e	Ataque	Acurácia conjunto D_{atk} (sem defesa)
MNIST	99.40%	FGSM	59.65%
		BIM	63.50%
		<i>DeepFool</i>	0.60%
		CW	63.80%
CIFAR	89.76%	FGSM	40.60%
		BIM	49.55%
		<i>DeepFool</i>	42.70%
		CW	5.00%

TAB. 3: O conjunto de hiperparâmetros c_b definido para o *MagNet* e *MultiMagNet* em cada algoritmo de ataque.

<i>Dataset</i>	Ataque	Taxa de Falsos Positivos T_{fp_b}	Abordagem do Limiar δ_b^{23}	Métrica me_b	Temperatura k_b
MNIST	FGSM - BIM	0.001	<i>MTA</i>	RE	1
	<i>DeepFool</i>	0.07	<i>MTA</i>	JSD	1
	CW	0.05	<i>minMTA</i>	RE	1
CIFAR-10	FGSM - BIM	0.1	<i>minMTA</i>	JSD	15
	<i>DeepFool</i>	0.07	<i>minMTA</i>	JSD	1
	CW	0.07	<i>minMTA</i>	JSD	5

5.5.1 CONJUNTO DE EXPERIMENTOS I

O objetivo principal do Conjunto de Experimentos I foi avaliar as capacidades de detecção do *MagNet* e *MultiMagNet*. Para isso, foi utilizado o conjunto de teste D , de acordo com o seguinte critério: a cada 100 imagens de entrada, oriundas do conjunto D , um novo comitê R , contendo n *autoencoders*, foi formado aleatoriamente, sendo n um número ímpar, tal que $n \in \{3, 5, 7, 9\}$ para o *MultiMagNet* e $n = 1$ para o *MagNet*. Uma vez que o conjunto D contém 4.000 imagens, sendo 2.000 delas rotuladas como legítimas e 2.000 rotuladas como contraditórias, foram realizados ao todo 40 experimentos. Em cada experimento, foram calculadas as métricas Acurácia (ACC), Valor Preditivo Positivo (VPP) e Valor Preditivo Negativo (VPN), que estão definidas pela Equação 12. As Tabelas 4 e 5 apresentam a média μ e o desvio padrão σ dos 40 resultados obtidos com o cálculo das métricas ACC, VPP e VPN nos *datasets* MNIST e CIFAR-10, respectivamente.

Ao analisar os resultados das camadas de detecção do *MagNet* e *MultiMagNet*, presentes nas Tabelas 4 e 5, fica claro que o aumento do efeito não determinístico, proporcionado pela formação aleatória de comitês de *autoencoders*, contribuiu para que o *MultiMagNet* apresentasse melhores resultados na métrica *VPN*, em comparação ao *MagNet*, para am-

TAB. 4: Média e desvio padrão dos resultados obtidos pelo *MagNet* e *MultiMagNet* em 40 experimentos no *dataset* MNIST.

Método de Defesa	Algoritmo de Ataque	ACC ($\mu - \sigma$) (%)	VPP ($\mu - \sigma$) (%)	VPN ($\mu - \sigma$) (%)
<i>MagNet</i>	FGSM	99.90 - 0.37	99.81 - 0.70%	100.00 - 0.00
	BIM	99.90 - 0.30	99.81 - 0.58	100.00 - 0.00
	<i>DeepFool</i>	96.43 - 2.20	92.83 - 4.30	100.00 - 0.00
	CW	56.00 - 5.17	94.84 - 2.99	27.54 - 7.85
<i>MultiMagNet</i>	FGSM	99.92 - 0.26	99.85 - 0.51	100.00 - 0.00
	BIM	99.92 - 0.26	99.85 - 0.54	100.00 - 0.00
	<i>DeepFool</i>	96.47 - 1.48	92.90 - 3.08	100.00 - 0.00
	CW	61.10 - 5.74	45.99 - 22.23	75.24 - 15.25

TAB. 5: Média e desvio padrão dos resultados obtidos pelo *MagNet* e *MultiMagNet* em 40 experimentos no *dataset* CIFAR-10.

Método de Defesa	Algoritmo de Ataque	ACC ($\mu - \sigma$) (%)	VPP ($\mu - \sigma$) (%)	VPN ($\mu - \sigma$) (%)
<i>MagNet</i>	FGSM	58.43 - 5.56	90.15 - 4.07	26.35 - 8.51
	BIM	77.72 - 3.91	88.98 - 3.50	69.75 - 6.42
	<i>DeepFool</i>	71.88 - 21.18	93.03 - 2.73	49.93 - 42.93
	CW	66.55 - 6.53	98.43 - 4.40	45.20 - 10.95
<i>MultiMagNet</i>	FGSM	65.88 - 4.48	67.13 - 12.64	65.06 - 13.22
	BIM	79.38 - 6.13	66.57 - 15.42	89.11 - 6.95
	<i>DeepFool</i>	90.25 - 4.66	80.67 - 9.22	99.86 - 0.69
	CW	68.95 - 4.45	48.82 - 8.80	84.31 - 5.96

bos os *datasets*. Os bons resultados apresentados pelo *MultiMagNet* na métrica *VPN* apontam para sua melhor capacidade na detecção de imagens contraditórias, como pode ser visto, principalmente, pelos resultados de ambos os métodos de defesa para os algoritmos de ataque BIM e *DeepFool*, presentes na Tabela 5.

Entretanto, também é possível perceber que o *MultiMagNet* apresentou resultados inferiores ao *MagNet* na métrica *VPP*, para o ataque CW aplicado no *dataset* MNIST (veja a Tabela 4), e para os quatro algoritmos de ataque aplicados no *dataset* CIFAR-10 (veja a Tabela 5). A queda dos resultados do *MultiMagNet* para a métrica *VPP* pode ter sido influenciada pelo valor Abordagem de Limiar Mínimo (*minTA*), que foi definido pela Etapa de Calibração do *MultiMagNet* para o hiperparâmetro **Abordagem do Limiar** nos respectivos cenários de ataque, como mostrado pela Tabela 3. Como já mencionado, o valor *minTA* atribui o menor limiar calculado para todos os n *autoencoders* do comitê R , o que pode ter contribuído para o *MultiMagNet* ter classificado mais imagens legítimas como contraditórias. Além do valor *minTA*, os valores definidos para o hiperparâmetro T_{fp} também podem ter influenciado na queda dos resultados do *MultiMagNet* na métrica *VPP*, uma vez que valores de T_{fp} menores dos que os valores presentes na Tabela 3 produziram um aumento na métrica *VPP*, em detrimento da métrica *VPN*.

Contudo, vale ressaltar que a queda na métrica *VPP* é justificável, pelo fato do *Mul-*

tiMagNet ter apresentado uma maior acurácia que o *MagNet* em cada um dos quatro algoritmos de ataque, quando aplicados em ambos os *datasets*. Tal diferença na acurácia alcançou 18,37 pontos percentuais para o algoritmo de ataque *DeepFool* no *dataset* CIFAR-10 (veja a Tabela 3). Outro ponto importante a ser destacado nas Tabelas 4 e 5 são os menores valores de desvio padrão, que, em suma, revelam uma maior confiabilidade e estabilidade do *MultiMagNet* em detrimento do *MagNet*. Todos esses pontos levantados apontam para a validade da hipótese levantada por este trabalho.

5.5.2 CONJUNTO DE EXPERIMENTOS II

O principal objetivo do Conjunto de Experimentos II, por sua vez, foi avaliar a capacidade dos métodos de defesa *MagNet* e *MultiMagNet* em proteger os modelos de classificação. Para isso, o Conjunto de Experimentos II foi dividido em cinco Cenários que verificaram a acurácia do classificação da aplicação no conjunto D : (i) sem nenhuma defesa, (ii) com o *MagNet* sem o Reformador presente, (iii) com o *MultiMagNet* sem o Reformador presente, (iv) com o *MagNet* e o Reformador presentes e, por fim, (v) com o *MultiMagNet* e o Reformador presentes. Os resultados correspondentes a cada Cenário estão organizados na Tabela 6.

Os resultados referentes ao Cenário (i) serviram de controle para os demais Cenários. Com base nos resultados do Cenário (i) foi possível mensurar o quanto os métodos de defesa foram, de fato, eficazes em proteger o classificador da aplicação. Já os Cenários (ii) e (iii) verificaram, respectivamente, a influência das camadas de detecção do *MagNet* e *MultiMagNet* em proteger o classificador da aplicação. Por fim, os Cenários (iv) e (v) verificaram se as camadas de reforma do *MagNet* e *MultiMagNet* realmente atuaram como uma segunda linha de defesa, removendo as perturbações das imagens contraditórias que não foram identificadas pela camada de detecção. De forma semelhante à metodologia empregada no CE I, foram realizadas novas formações aleatórias de comitês R a cada 100 imagens de entrada do conjunto D , resultando em um total de 40 experimentos.

Ao analisar os resultados correspondentes aos Cenários (i) e (ii), presentes na Tabela 6, percebe-se que a presença do *MagNet* forneceu alguma proteção ao classificador da aplicação F . Entretanto, se os resultados referentes ao Cenário (ii) forem comparados com os resultados referentes ao Cenário (iii), fica notório que o *MultiMagNet* também conseguiu superar o *MagNet* em proteger o classificador contra as imagens contraditórias geradas pelos quatro algoritmos de ataque, com destaque para o *DeepFool* e o CW quando aplicados no *dataset* CIFAR-10. Vale mencionar também que a camada de reforma também contribuiu de maneira significativa para o aumento da acurácia do classificador da aplica-

TAB. 6: Acurácias obtidas pelo classificador da aplicação F quando avaliado no conjunto D em cinco Cenários de ataque diferentes.

<i>Dataset</i>	Ataque	Sem defesa ($\mu - \sigma$) (%)	<i>MagNet</i> sem o Reformador ($\mu - \sigma$) (%)	<i>MultiMagNet</i> sem o Reformador ($\mu - \sigma$) (%)	<i>MagNet</i> ($\mu - \sigma$) (%)	<i>MultiMagNet</i> ($\mu - \sigma$) (%)
MNIST	FGSM	79.53 - 4.52	99.37 - 1.11	99.46 - 1.06	98.98 - 1.75	99.17 - 1.41
	BIM	81.45 - 3.29	99.40 - 1.03	99.49 - 1.20	99.16 - 1.18	99.19 - 1.61
	<i>DeepFool</i>	50.00 - 5.38	99.95 - 0.31	99.95 - 0.31	99.90 - 0.46	99.95 - 0.31
	CW	81.60 - 4.33	92.92 - 3.21	98.71 - 1.72	96.19 - 2.20	98.89 - 1.36
CIFAR-10	FGSM	64.83 - 2.94	67.27 - 1.01	72.81 - 3.59	70.08 - 2.49	72.91 - 2.75
	BIM	69.30 - 1.85	89.12 - 2.92	89.94 - 1.42	92.58 - 2.41	96.75 - 3.54
	<i>DeepFool</i>	65.97 - 2.03	79.12 - 10.05	92.52 - 1.67	87.15 - 4.48	92.91 - 2.88
	CW	47.03 - 1.29	59.14 - 3.29	71.86 - 5.74	60.08 - 3.27	72.36 - 5.22

ção, como pode ser visto, principalmente, nos resultados correspondentes ao Cenário (v). O motivo da camada de reforma ter melhorado os resultados das camadas de detecção do *MultiMagNet* pode estar relacionado ao fato de, mesmo com a formação aleatória de comitês em tempo execução, algumas imagens contraditórias ainda foram capazes de chegarem ao reformador sem serem detectadas pelo comitê. Diante disso, é possível deduzir pelos resultados referentes ao Cenário (v) que a camada de reforma realmente agiu como uma segunda linha de defesa, impedindo que as perturbações presentes em algumas imagens contraditórias afetassem negativamente o classificador da aplicação.

De maneira geral, os resultados presentes nas Tabelas 4, 5 e 6 enfatizam que o *MultiMagNet* foi mais eficaz que o *MagNet* em detectar imagens contraditórias e proteger o classificador da aplicação, fornecendo mais evidências significativas de que a hipótese levantada inicialmente por esta dissertação, é válida.

6 CONSIDERAÇÕES FINAIS

Nos últimos anos, os algoritmos de Aprendizado Profundo, em especial as Redes Neurais Convolucionais, vêm evoluindo rapidamente, ao ponto de superarem um humano em tarefas de reconhecimento e classificação de imagens. Por esse motivo, diversas aplicações críticas em segurança vêm utilizando as RNCs como principal solução de tarefas como: automação de veículos (BOJARSKI et al., 2016), reconhecimento biométrico (MIDDLEHURST, 2015; BAE et al., 2018), sistemas de vigilância (DING et al., 2018) e identificação de dígitos manuscritos (TOLOSANA et al., 2018; SRIVASTAVA et al., 2019).

Contudo, diversos trabalhos vêm demonstrando que as RNCs podem ser propositalmente induzidas a cometer erros de classificação diante de imagens contraditórias (*i.e.*, imagens que contenham perturbações geradas por algoritmos de ataques maliciosos), fato que impossibilita a aplicação desses algoritmos de aprendizado em vários sistemas de apoio à decisão (KLARREICH, 2016). Embora várias defesas tenham sido criadas para detectar imagens contraditórias, a maioria tem sido superada por permitir que o atacante mapeie facilmente o seu comportamento. A fim de evitar a previsibilidade de comportamento, uma iniciativa de pesquisa buscou criar um método de detecção não determinístico chamado *MagNet* (MENG; CHEN, 2017). Estudos recentes revelam que, apesar do não determinismo, o *MagNet* ainda pode ser superado por ataques contraditórios, principalmente por escolher aleatoriamente apenas um componente de defesa em tempo de execução (CARLINI; WAGNER, 2017b).

Diante disso, o presente trabalho levantou a hipótese de que a introdução de múltiplos componentes de defesa, selecionados aleatoriamente, no método de detecção pode ampliar o efeito do não determinismo de maneira a torná-lo mais robusto que o *MagNet* frente aos diferentes tipos de ataque. Assim, o presente trabalho introduziu o *MultiMagNet*, um método de detecção de imagens contraditórias que utiliza múltiplos componentes de defesa, implementados como *autoencoders*, selecionados aleatoriamente e organizados em comitês para tomada de decisão. Os resultados de diversos experimentos em imagens dos *datasets* MNIST e CIFAR-10 apontaram para a validade da hipótese levantada, ao mostrar que o *MultiMagNet* superou o *MagNet*, sua versão não determinística que utiliza apenas um *autoencoder*, na maioria dos cenários avaliados, sendo capaz de detectar e proteger classificadores contra perturbações geradas por diferentes algoritmos de ataque.

Em suma, as principais contribuições alcançadas por esta dissertação são:

- a) a melhora da capacidade de proteção de classificadores em relação aos métodos de defesa do estado da arte, com a elaboração do *MultiMagNet*: uma arquitetura não determinística baseada na escolha de múltiplos componentes de defesa contra imagens contraditórias;
- b) a realização de um estudo comparativo entre o *MultiMagNet* e o método de defesa *MagNet*, proposto por Meng e Chen (2017);
- c) A elaboração de uma taxonomia para a organização dos métodos de defesa contra imagens contraditórias;
- d) A elaboração de novas taxonomias para os diferentes tipos de imagens contraditórias e ataques com imagens contraditórias, baseadas nos trabalhos de Yuan et al. (2017b), Akhtar e Mian (2018), Barreno et al. (2010), Xiao (2017), Kumar e Mehta (2017), Brendel e Bethge (2017) e Goodfellow et al. (2015).

Como iniciativas para trabalhos futuros, podem ser listadas, as seguintes tarefas: (i) a implementação de algoritmos para otimização dos hiperparâmetros e limiares do *MultiMagNet*; (ii) a implementação e avaliação de uma arquitetura híbrida não determinística, que contenha duas camadas de defesa: a camada reativa, onde seja escolhido um ou mais métodos de defesa reativos para detecção de imagens contraditórias, e uma camada proativa, onde seja escolhido um ou mais modelos proativos para a classificação das imagens oriundas da camada reativa; (iii) a avaliação do *MultiMagNet* em outros *datasets* e, por fim, (iv) a investigação de diferentes técnicas de pré-processamento para tratamento das imagens contraditórias.

7 REFERÊNCIAS BIBLIOGRÁFICAS

1. ABBASI, M.; GAGNÉ, C. Robustness to adversarial examples through an ensemble of specialists. **arXiv preprint arXiv:1702.06856**, . , 2017.
2. AKHTAR, N.; MIAN, A. Threat of adversarial attacks on deep learning in computer vision: A survey. **arXiv preprint arXiv:1801.00553**, . , 2018.
3. BAE, G.; LEE, H.; SON, S.; HWANG, D. ; KIM, J. Secure and robust user authentication using partial fingerprint matching. In: CONSUMER ELECTRONICS (ICCE), 2018 IEEE INTERNATIONAL CONFERENCE ON, , 2018. **Anais...** [S.l.: s.n.], 2018, p. 1–6.
4. BALUJA, S.; FISCHER, I. Adversarial transformation networks: Learning to generate adversarial examples. **arXiv preprint arXiv:1703.09387**, . , 2017.
5. BARRENO, M.; NELSON, B.; JOSEPH, A. D. ; TYGAR, J. D. The security of machine learning. **Machine Learning**, v. 81, n. 2, p. 121–148, 2010.
6. BOJARSKI, M.; DEL TESTA, D.; DWORAKOWSKI, D.; FIRNER, B.; FLEPP, B.; GOYAL, P.; JACKEL, L. D.; MONFORT, M.; MULLER, U.; ZHANG, J. ; OTHERS. End to end learning for self-driving cars. **arXiv preprint arXiv:1604.07316**, . , 2016.
7. BRENDDEL, W.; BETHGE, M. Comment on Biologically inspired protection of deep networks from adversarial attacks. , , 2017. Disponível em: <<http://arxiv.org/abs/1704.01547>>. .
8. BRENDDEL, W.; RAUBER, J. ; BETHGE, M. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. **arXiv preprint arXiv:1712.04248**, . , 2017.
9. BROWN, T. B.; MANÉ, D.; ROY, A.; ABADI, M. ; GILMER, J. Adversarial patch. **arXiv preprint arXiv:1712.09665**, . , 2017.
10. BROWNLEE, J. What is deep learning?. , , 2016. Disponível em: <<https://machinelearningmastery.com/what-is-deep-learning/>>. Acesso em: 29 de janeiro de 2019.
11. CAO, X.; GONG, N. Z. Mitigating evasion attacks to deep neural networks via region-based classification. In: PROCEEDINGS OF THE 33RD ANNUAL COMPUTER SECURITY APPLICATIONS CONFERENCE, , ACSAC 2017, . , 2017. **Anais...** New York, NY, USA: ACM, 2017, p. 278–287. Disponível em: <<http://doi.acm.org/10.1145/3134600.3134606>>. .
12. CARLINI, N.; WAGNER, D. Adversarial examples are not easily detected: Bypassing ten detection methods. In: PROCEEDINGS OF THE 10TH ACM WORKSHOP ON ARTIFICIAL INTELLIGENCE AND SECURITY, , 2017. **Anais...** [S.l.: s.n.], 2017, p. 3–14.

13. CARLINI, N.; WAGNER, D. Magnet and "efficient defenses against adversarial attacks" are not robust to adversarial examples. **arXiv preprint arXiv:1711.08478**, . , 2017.
14. CARLINI, N.; WAGNER, D. Towards evaluating the robustness of neural networks. In: 2017 IEEE SYMPOSIUM ON SECURITY AND PRIVACY (SP), , 2017. **Anais...** [S.l.: s.n.], 2017, p. 39–57.
15. CARRARA, F.; FALCHI, F.; CALDELLI, R.; AMATO, G. ; BECARELLI, R. Adversarial image detection in deep neural networks. **Multimedia Tools and Applications**, , p. 1–21, 2018.
16. CHAKRABORTY, A.; ALAM, M.; DEY, V.; CHATTOPADHYAY, A. ; MUKHOPADHYAY, D. Adversarial attacks and defences: A survey. **arXiv preprint arXiv:1810.00069**, . , 2018.
17. CHEN, J.; MENG, Z.; SUN, C.; TANG, W. ; ZHU, Y. Reabsnet: Detecting and revising adversarial examples. **arXiv preprint arXiv:1712.08250**, . , 2017.
18. CHEN, P.-Y.; ZHANG, H.; SHARMA, Y.; YI, J. ; HSIEH, C.-J. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. **10th ACM Workshop on Artificial Intelligence and Security (AISEC)**, . , 2017.
19. CHENG, J.-Z.; NI, D.; CHOU, Y.-H.; QIN, J.; TIU, C.-M.; CHANG, Y.-C.; HUANG, C.-S.; SHEN, D. ; CHEN, C.-M. Computer-aided diagnosis with deep learning architecture: applications to breast lesions in us images and pulmonary nodules in ct scans. **Scientific reports**, v. 6, p. 24454, 2016.
20. CISSE, M.; BOJANOWSKI, P.; GRAVE, E.; DAUPHIN, Y. ; USUNIER, N. Parseval Networks: Improving Robustness to Adversarial Examples. , , 2017. Disponível em: <<http://arxiv.org/abs/1704.08847>>. .
21. DAHL, G. E.; STOKES, J. W.; DENG, L. ; YU, D. Large-scale malware classification using random projections and neural networks. In: 2013 IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING, , 2013. **Anais...** [S.l.: s.n.], 2013, p. 3422–3426.
22. DAHL, G. E.; YU, D.; DENG, L. ; ACERO, A. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. **Trans. Audio, Speech and Lang. Proc.**, v. 20, n. 1, p. 30–42, 2012. Disponível em: <<http://dx.doi.org/10.1109/TASL.2011.2134090>>. .
23. DING, L.; FANG, W.; LUO, H.; LOVE, P. E.; ZHONG, B. ; OUYANG, X. A deep hybrid learning model to detect unsafe behavior: integrating convolution neural networks and long short-term memory. **Automation in Construction**, v. 86, p. 118–124, 2018.
24. EVTIMOV, I.; EYKHOLT, K.; FERNANDES, E.; KOHNO, T.; LI, B.; PRAKASH, A.; RAHMATI, A. ; SONG, D. Robust Physical-World Attacks on Deep Learning Models. In: PROCEEDINGS OF THE 2016 ACM SIGSAC CONFERENCE ON COMPUTER AND COMMUNICATIONS SECURITY, , 2017. **Anais...** [S.l.: s.n.], 2017, p. 1528–1540. Disponível em: <<http://arxiv.org/abs/1707.08945>>. .

25. FACELI, K.; LORENA, A.; GAMA, J. ; CARVALHO, A. **Inteligência Artificial—uma abordagem de aprendizado de máquina**. [S.l.]: Rio de Janeiro: LTC, 2011.
26. FEINMAN, R.; CURTIN, R. R.; SHINTRE, S. ; GARDNER, A. B. Detecting Adversarial Samples from Artifacts. , , 2017. Disponível em: <<http://arxiv.org/abs/1703.00410>>. .
27. GERSHGORN, D. Fooling the machine: The byzantine science of deceiving artificial intelligence. **Popular Science, March**, v. 30. , 2016.
28. GONG, Z.; WANG, W. ; KU, W.-S. Adversarial and clean data are not twins. **arXiv preprint arXiv:1704.04960**, , , 2017.
29. GOODFELLOW, I.; BENGIO, Y. ; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016.
30. GOODFELLOW, I.; PAPERNOT, N. Is attacking machine learning easier than defending it?. **Cleverhans Blog**, , 2017. Disponível em: <<http://www.cleverhans.io/security/privacy/ml/2017/02/15/why-attacking-machine-learning-is-easier-than-defending-it.html>>. Acesso em: 15 de março de 2019.
31. GOODFELLOW, I.; POUGET-ABADIE, J.; MIRZA, M.; XU, B.; WARDE-FARLEY, D.; OZAIR, S.; COURVILLE, A. ; BENGIO, Y. Generative adversarial nets. In: **ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS**, , 2014. **Anais...** [S.l.: s.n.], 2014, p. 2672–2680.
32. GOODFELLOW, I.; SHLENS, J. ; SZEGEDY, C. Explaining and harnessing adversarial examples. In: **INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS**, , 2015. **Anais...** [S.l.: s.n.], 2015. Disponível em: <<http://arxiv.org/abs/1412.6572>>. .
33. GROSSE, K.; MANOHARAN, P.; PAPERNOT, N.; BACKES, M. ; MCDANIEL, P. On the (Statistical) Detection of Adversarial Examples. , , 2017. Disponível em: <<http://arxiv.org/abs/1702.06280>>. .
34. GU, S.; RIGAZIO, L. Towards Deep Neural Network Architectures Robust to Adversarial Examples. , , 2014. Disponível em: <<http://arxiv.org/abs/1412.5068>>. .
35. GUO, Y.; LIU, Y.; OERLEMANS, A.; LAO, S.; WU, S. ; LEW, M. S. Deep learning for visual understanding: A review. **Neurocomputing**, v. 187, p. 27–48, 2016.
36. HE, K.; ZHANG, X.; REN, S. ; SUN, J. Deep Residual Learning for Image Recognition. In: **2016 IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR)**, , 2016. **Anais...** [S.l.]: IEEE, 2016, p. 770–778. Disponível em: <<http://ieeexplore.ieee.org/document/7780459/>>. .
37. HE, W.; WEI, J.; CHEN, X.; CARLINI, N. ; SONG, D. Adversarial Example Defenses: Ensembles of Weak Defenses are not Strong. In: **11TH USENIX WORKSHOP ON OFFENSIVE TECHNOLOGIES (WOOT' 17)**, , 2017. **Anais...** Vancouver, CA: [s.n.], 2017. Disponível em: <<https://www.usenix.org/system/files/conference/woot17/woot17-paper-he.pdf>>. .

38. HEATON, J.; POLSON, N. ; WITTE, J. H. Deep learning for finance: deep portfolios. **Applied Stochastic Models in Business and Industry**, v. 33, n. 1, p. 3–12, 2017.
39. HENDRYCKS, D.; GIMPEL, K. Early methods for detecting adversarial images. **Workshop track -ICLR 2017**, . , 2017.
40. HINTON, G.; VINYALS, O. ; DEAN, J. Distilling the knowledge in a neural network. **Deep Learning and Representation Learning Workshop at NIPS 2014. arXiv preprint arXiv:1503.02531**, . , 2015.
41. HU, J.; SHEN, L. ; SUN, G. Squeeze-and-excitation networks. **arXiv preprint arXiv:1709.01507**, . , 2017.
42. HUANG, L.; JOSEPH, A. D.; NELSON, B.; RUBINSTEIN, B. I. ; TYGAR, J. Adversarial machine learning. In: PROCEEDINGS OF THE 4TH ACM WORKSHOP ON SECURITY AND ARTIFICIAL INTELLIGENCE, , 2011. **Anais...** [S.l.: s.n.], 2011, p. 43–58.
43. HUANG, R.; XU, B.; SCHUURMANS, D. ; SZEPESVÁRI, C. Learning with a strong adversary. **arXiv preprint arXiv:1511.03034**, . , 2015.
44. KANNAN, H.; KURAKIN, A. ; GOODFELLOW, I. Adversarial logit pairing. **arXiv preprint arXiv:1803.06373**, . , 2018.
45. KARMON, D.; ZORAN, D. ; GOLDBERG, Y. Lavan: Localized and visible adversarial noise. **arXiv preprint arXiv:1801.02608**, . , 2018.
46. KARPATY, A. What I learned from competing against a ConvNet on Imagenet. **2015-01-24**, , 2014. Disponível em: <<http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet>>. Acesso em: 15 de janeiro de 2019.
47. KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. In: INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS (ICLR), , 2015. **Anais...** [S.l.: s.n.], 2015. .
48. KLARREICH, E. Learning securely. **Communications of the ACM**, v. 59, n. 11, p. 12–14, 2016.
49. KNORR, E. How paypal beats the bad guys with machine learning. **InfoWorld**, , 2015. Disponível em: <<https://www.infoworld.com/article/2907877/machine-learning/how-paypal-reduces-fraud-with-machine-learning.html>>. .
50. KRIZHEVSKY, A.; HINTON, G. Learning multiple layers of features from tiny images. . , , 2009.
51. KRIZHEVSKY, A.; SUTSKEVER, I. ; HINTON, G. E. Imagenet classification with deep convolutional neural networks. , , p. 1097–1105, 2012.
52. KUMAR, A.; MEHTA, S. A Survey on Resilient Machine Learning. , , 2017. Disponível em: <<https://arxiv.org/ftp/arxiv/papers/1707/1707.03184.pdf>>. .
53. KURAKIN, A.; GOODFELLOW, I. ; BENGIO, S. Adversarial examples in the physical world. **arXiv preprint arXiv:1607.02533**, . , 2016.

54. KURAKIN, A.; GOODFELLOW, I. ; BENGIO, S. Adversarial machine learning at scale. **arXiv preprint arXiv:1611.01236**, . , 2016.
55. LECUN, Y.; BOTTOU, L.; BENGIO, Y. ; HAFFNER, P. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278–2324, 1998.
56. LECUN, Y.; KAVUKCUOGLU, K. ; FARABET, C. Convolutional Networks and Applications in Vision. In: PROCEEDINGS OF 2010 IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, , 2010. **Anais...** [S.l.]: IEEE, 2010, p. 253–256. Disponível em: <ieeexplore.ieee.org/document/5537907/>. .
57. LI, X.; LI, F. Adversarial examples detection in deep networks with convolutional filter statistics. **arXiv preprint arXiv:1612.07767**, . , 2016.
58. LITJENS, G.; KOOI, T.; BEJNORDI, B. E.; SETIO, A. A. A.; CIOMPI, F.; GHAFOORIAN, M.; VAN DER LAAK, J. A.; VAN GINNEKEN, B. ; SÁNCHEZ, C. I. A survey on deep learning in medical image analysis. **Medical image analysis**, v. 42, p. 60–88, 2017.
59. LIU, W.; WANG, Z.; LIU, X.; ZENG, N.; LIU, Y. ; ALSAADI, F. E. A survey of deep neural network architectures and their applications. **Neurocomputing**, v. 234, n. November 2016, p. 11–26, 2017. Disponível em: <http://dx.doi.org/10.1016/j.neucom.2016.12.038>. .
60. LIU, X.; CHENG, M.; ZHANG, H. ; HSIEH, C.-J. Towards robust neural networks via random self-ensemble. **arXiv preprint arXiv:1712.00673**, . , 2017.
61. MADRY, A.; MAKELOV, A.; SCHMIDT, L.; TSIPRAS, D. ; VLADU, A. Towards deep learning models resistant to adversarial attacks. **arXiv preprint arXiv:1706.06083**, . , 2017.
62. MAHAPATRA, S. Why deep learning over traditional machine learning?. , , 2018. Disponível em: <https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063>. Acesso em: 21 de março de 2019.
63. MENG, D.; CHEN, H. Magnet: a two-pronged defense against adversarial examples. In: PROCEEDINGS OF THE 2017 ACM SIGSAC CONFERENCE ON COMPUTER AND COMMUNICATIONS SECURITY, , 2017. **Anais...** [S.l.: s.n.], 2017, p. 135–147.
64. METZEN, J. H.; GENEWEIN, T.; FISCHER, V. ; BISCHOFF, B. On detecting adversarial perturbations. **arXiv preprint arXiv:1702.04267**, . , 2017.
65. METZEN, J. H.; KUMAR, M. C.; BROX, T. ; FISCHER, V. Universal adversarial perturbations against semantic image segmentation. **stat**, v. 1050, p. 19, 2017.
66. MIDDLEHURST, C. China unveils world’s first facial recognition atm. , , 2015. Disponível em: <https://www.telegraph.co.uk/news/worldnews/asia/china/11643314/China-unveils-worlds-first-facial-recognition-ATM.html>. Acesso em: 21 de janeiro de 2019.

67. MOOSAVI-DEZFOOLI, S.-M.; FAWZI, A.; FAWZI, O.; FROSSARD, P. ; SOATTO, S. Analysis of universal adversarial perturbations. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), , 2017. **Anais...** [S.l.: s.n.], 2017. Disponível em: <<http://arxiv.org/abs/1705.09554>>. .
68. MOOSAVI-DEZFOOLI, S.-M.; FAWZI, A. ; FROSSARD, P. Deepfool: a simple and accurate method to fool deep neural networks. In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, , 2016. **Anais...** [S.l.: s.n.], 2016, p. 2574–2582.
69. NGUYEN, A.; YOSINSKI, J. ; CLUNE, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, v. 07-12-June, p. 427–436, 2015.
70. NICOLAE, M.-I.; SINN, M.; TRAN, M. N.; RAWAT, A.; WISTUBA, M.; ZAN-TEDESCHI, V.; BARACALDO, N.; CHEN, B.; LUDWIG, H.; MOLLOY, I. ; EDWARDS, B. Adversarial robustness toolbox v0.3.0. **CoRR**, v. 1807.01069, 2018. Disponível em: <<https://arxiv.org/pdf/1807.01069>>. .
71. PAPERNOT, N. **Characterizing the Limits and Defenses of Machine Learning in Adversarial Settings**. 2018. Tese () – The Pennsylvania State University - Computer Science and Engineering, , 2018. Disponível em: <<https://etda.libraries.psu.edu/catalog/15065ngp5056>>. Acesso em: 30 de janeiro de 2019.
72. PAPERNOT, N. A marauder’s map of security and privacy in machine learning. **arXiv preprint arXiv:1811.01134**, . , 2018.
73. PAPERNOT, N.; MCDANIEL, P. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. **arXiv preprint arXiv:1803.04765**, . , 2018.
74. PAPERNOT, N.; MCDANIEL, P.; GOODFELLOW, I.; JHA, S.; CELIK, Z. B. ; SWAMI, A. Practical Black-Box Attacks against Machine Learning. In: ACM ASIA CONFERENCE ON COMPUTER AND COMMUNICATIONS SECURITY (ASIACCS), , 2017. **Anais...** [S.l.: s.n.], 2017, p. 506–519. Disponível em: <<http://arxiv.org/abs/1602.02697>>. .
75. PAPERNOT, N.; MCDANIEL, P.; JHA, S.; FREDRIKSON, M.; CELIK, Z. B. ; SWAMI, A. The limitations of deep learning in adversarial settings. In: SECURITY AND PRIVACY (EUROS&P), 2016 IEEE EUROPEAN SYMPOSIUM ON, , 2016. **Anais...** [S.l.: s.n.], 2016, p. 372–387.
76. PAPERNOT, N.; MCDANIEL, P.; SINHA, A. ; WELLMAN, M. Towards the science of security and privacy in machine learning. **arXiv preprint arXiv:1611.03814**, . , 2016.
77. PAPERNOT, N.; MCDANIEL, P.; WU, X.; JHA, S. ; SWAMI, A. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. In: PROCEEDINGS - 2016 IEEE SYMPOSIUM ON SECURITY AND PRIVACY, SP 2016, , 2016. **Anais...** [S.l.: s.n.], 2016, p. 582–597.

78. RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. ; OTHERS. Learning representations by back-propagating errors. **Cognitive modeling**, v. 5, n. 3, p. 1, 1988.
79. RUSSAKOVSKY, O.; DENG, J.; SU, H.; KRAUSE, J.; SATHEESH, S.; MA, S.; HUANG, Z.; KARPATY, A.; KHOSLA, A.; BERNSTEIN, M.; BERG, A. C. ; FEL-FEI, L. Imagenet large scale visual recognition challenge. **International Journal of Computer Vision (IJCV)**, v. 115, n. 3, p. 211–252, 2015.
80. SAK, H.; SENIOR, A. W. ; BEAUFAYS, F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In: PROCEEDINGS OF THE ANNUAL CONFERENCE OF INTERNATIONAL SPEECH COMMUNICATION ASSOCIATION (INTERSPEECH), , 2014. **Anais...** [S.l.: s.n.], 2014. .
81. SENGUPTA, S.; CHAKRABORTI, T. ; KAMBHAMPATI, S. Mtdeep: Securing deep neural nets against adversarial attacks with moving target defense. **arXiv preprint arXiv:1705.07213**, , 2017. Disponível em: <<http://arxiv.org/abs/1705.07213>>. .
82. SHEN, S.; JIN, G.; GAO, K. ; ZHANG, Y. Ape-gan: Adversarial perturbation elimination with gan, arxiv preprint. **arXiv preprint arXiv:1707.05474**, . , 2017.
83. SILVER, D.; HUANG, A.; MADDISON, C. J.; GUEZ, A.; SIFRE, L.; VAN DEN DRI-ESSCHE, G.; SCHRITTWIESER, J.; ANTONOGLU, I.; PANNEERSHELVAM, V.; LANCTOT, M.; DIELEMAN, S.; GREWE, D.; NHAM, J.; KALCHBRENNER, N.; SUTSKEVER, I.; LILICRAP, T.; LEACH, M.; KAVUKCUOGLU, K.; GRAE-PEL, T. ; HASSABIS, D. Mastering the game of Go with deep neural networks and tree search. **Nature**, v. 529, n. 7587, p. 484–489, 2016.
84. SILVER, D.; HUBERT, T.; SCHRITTWIESER, J.; ANTONOGLU, I.; LAI, M.; GUEZ, A.; LANCTOT, M.; SIFRE, L.; KUMARAN, D.; GRAE-PEL, T.; LILICRAP, T.; SIMONYAN, K. ; HASSABIS, D. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. **Science**, v. 362, n. 6419, p. 1140–1144, 2018. Disponível em: <<http://science.sciencemag.org/content/362/6419/1140>>. .
85. SPRINGENBERG, J. T.; DOSOVITSKIY, A.; BROX, T. ; RIEDMILLER, M. Striving for simplicity: The all convolutional net. **arXiv preprint arXiv:1412.6806**, . , 2014.
86. SRIVASTAVA, N.; HINTON, G. E.; KRIZHEVSKY, A.; SUTSKEVER, I. ; SALAKHUTDINOV, R. Dropout: a simple way to prevent neural networks from overfitting.. **Journal of machine learning research**, v. 15, n. 1, p. 1929–1958, 2014.
87. SRIVASTAVA, S.; PRIYADARSHINI, J.; GOPAL, S.; GUPTA, S. ; DAYAL, H. S. Optical character recognition on bank cheques using 2d convolution neural network. In: APPLICATIONS OF ARTIFICIAL INTELLIGENCE TECHNIQUES IN ENGI-NEERING, , 2019. **Anais...** [S.l.]: Springer, 2019, p. 589–596.
88. STRAUSS, T.; HANSELMANN, M.; JUNGINGER, A. ; ULMER, H. Ensemble methods as a defense to adversarial perturbations against deep neural networks. **arXiv preprint arXiv:1709.03423**, . , 2017.

89. SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S.; ANGUELOV, D.; ERHAN, D.; VANHOUCHE, V. ; RABINOVICH, A. Going deeper with convolutions. , , p. 1–9, 2015.
90. SZEGEDY, C.; VANHOUCHE, V.; IOFFE, S.; SHLENS, J. ; WOJNA, Z. Rethinking the Inception Architecture for Computer Vision. In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, , 2015. **Anais...** [S.l.: s.n.], 2015, p. 2818–2826. Disponível em: <<http://arxiv.org/abs/1512.00567>>. .
91. SZEGEDY, C.; ZAREMBA, W.; SUTSKEVER, I.; BRUNA, J.; ERHAN, D.; GOODFELLOW, I. ; FERGUS, R. Intriguing properties of neural networks. In: INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS, , 2013. **Anais...** [S.l.: s.n.], 2013, p. 1–10. Disponível em: <<http://arxiv.org/abs/1312.6199>>. .
92. TOLOSANA, R.; VERA-RODRIGUEZ, R.; FIERREZ, J. ; ORTEGA-GARCIA, J. Exploring recurrent neural networks for on-line handwritten signature biometrics. **IEEE Access**, v. 6, n. 5128-5138, p. 1–7, 2018.
93. TRAMÈR, F.; KURAKIN, A.; PAPERNOT, N.; BONEH, D. ; MCDANIEL, P. Ensemble Adversarial Training: Attacks and Defenses. , , p. 1–15, 2017. Disponível em: <<http://arxiv.org/abs/1705.07204>>. .
94. TRAMÈR, F.; PAPERNOT, N.; GOODFELLOW, I.; BONEH, D. ; MCDANIEL, P. The space of transferable adversarial examples. **arXiv preprint arXiv:1704.03453**, . , 2017.
95. VOROBEYCHIK, Y.; KANTARCIOGLU, M. Adversarial machine learning. **Synthesis Lectures on Artificial Intelligence and Machine Learning**, v. 12, n. 3, p. 1–169, 2018.
96. WU, Y.; SCHUSTER, M.; CHEN, Z.; LE, Q. V.; NOROUZI, M.; MACHEREY, W.; KRIKUN, M.; CAO, Y.; GAO, Q.; MACHEREY, K.; KLINGNER, J.; SHAH, A.; JOHNSON, M.; LIU, X.; KAISER, L.; GOUWS, S.; KATO, Y.; KUDO, T.; KAZAWA, H.; STEVENS, K.; KURIAN, G.; PATIL, N.; WANG, W.; YOUNG, C.; SMITH, J.; RIESA, J.; RUDNICK, A.; VINYALS, O.; CORRADO, G.; HUGHES, M. ; DEAN, J. Google’s neural machine translation system: Bridging the gap between human and machine translation. **CoRR**, v. abs/1609.08144, 2016. Disponível em: <<http://arxiv.org/abs/1609.08144>>. .
97. XIAO, H. **Adversarial and Secure Machine Learning**. 2017. Tese () – Universität München, , 2017. Disponível em: <<https://mediatum.ub.tum.de/1335448>>. Acesso em: 04 de fevereiro de 2019.
98. XIE, C.; WANG, J.; ZHANG, Z.; REN, Z. ; YUILLE, A. Mitigating adversarial effects through randomization. **arXiv preprint arXiv:1711.01991**, . , 2017.
99. XU, W.; EVANS, D. ; QI, Y. Feature squeezing: Detecting adversarial examples in deep neural networks. **Network and Distributed Systems Security Symposium (NDSS) 2018**, . , 2018.

100. YADAV, S.; SUBRAMANIAN, S. Detection of application layer ddos attack by feature learning using stacked autoencoder. In: 2016 INTERNATIONAL CONFERENCE ON COMPUTATIONAL TECHNIQUES IN INFORMATION AND COMMUNICATION TECHNOLOGIES (ICCTICT), , 2016. **Anais...** [S.l.: s.n.], 2016, p. 361–366.
101. YUAN, X.; LI, C. ; LI, X. Deepdefense: Identifying ddos attack via deep learning. In: 2017 IEEE INTERNATIONAL CONFERENCE ON SMART COMPUTING (SMARTCOMP), , 2017. **Anais...** [S.l.: s.n.], 2017, p. 1–8.
102. YUAN, X.; HE, P.; ZHU, Q.; BHAT, R. R. ; LI, X. Adversarial examples: Attacks and defenses for deep learning. **arXiv preprint arXiv:1712.07107**, , 2017.
103. ZANTEDESCHI, V.; NICOLAE, M.-I. ; RAWAT, A. Efficient defenses against adversarial attacks. In: PROCEEDINGS OF THE 10TH ACM WORKSHOP ON ARTIFICIAL INTELLIGENCE AND SECURITY, , 2017. **Anais...** [S.l.: s.n.], 2017, p. 39–49.
104. ZANTEDESCHI, V.; NICOLAE, M.-I. ; RAWAT, A. Efficient defenses against adversarial attacks. In: PROCEEDINGS OF THE 10TH ACM WORKSHOP ON ARTIFICIAL INTELLIGENCE AND SECURITY, , 2017. **Anais...** [S.l.: s.n.], 2017, p. 39–49.

8 APÊNDICES

APÊNDICE 1: ARQUITETURAS E HIPERPARÂMETROS DOS CLASSIFICADORES

TAB. 7: Arquitetura e hiperparâmetros definidos para o classificador da aplicação no *dataset* MNIST.

Camada		Hiperparâmetros
<i>Conv. ReLU</i>	$5 \times 5 \times 64$	Função de custo: entropia cruzada categórica
<i>Max Pooling</i>	2×2	Método otimizador: gradiente estocástico <i>Adam</i> [47]
<i>Conv. ReLU</i>	$3 \times 3 \times 128$	<i>batch size</i> = 128
<i>Max Pooling</i>	2×2	taxa de aprendizado = 0.01
<i>Dropout</i>	0.2	épocas = 50
<i>Conv. ReLU</i>	$3 \times 3 \times 128$	taxa de momento = 0.9
<i>Max Pooling</i>	2×2	-
<i>Dense ReLU</i>	128	
<i>Dense ReLU</i>	10	
<i>Softmax</i>	10	

TAB. 8: Arquitetura e hiperparâmetros definidos para o classificador da aplicação no *dataset* CIFAR-10.

Camada		Hiperparâmetros
<i>Conv. ReLU</i>	$3 \times 3 \times 96$ (<i>same</i>)	Função de custo: entropia cruzada categórica
<i>Conv. ReLU</i>	$3 \times 3 \times 96$ (<i>same</i>)	Algoritmo otimizador: GDE
<i>Conv. ReLU</i>	$3 \times 3 \times 96$ (<i>same</i>)	<i>batch size</i> = 128
<i>Max Pooling</i>	2×2	taxa de aprendizado = 0.01
<i>Dropout</i>	0.5	<i>decay</i> = 10^{-7}
<i>Conv. ReLU</i>	$3 \times 3 \times 192$ (<i>same</i>)	épocas = 350
<i>Conv. ReLU</i>	$3 \times 3 \times 192$ (<i>same</i>)	<i>Data augmentation:</i> <i>horizontal + vertical shift</i> = 0.1 <i>horizontal + vertical flip</i> = <i>true</i>
<i>Conv. ReLU</i>	$3 \times 3 \times 192$ (<i>same</i>)	-
<i>Max Pooling</i>	2×2	
<i>Dropout</i>	0.5	
<i>Conv. ReLU</i>	$3 \times 3 \times 192$ (<i>same</i>)	
<i>Conv. ReLU</i>	$1 \times 3 \times 192$ (<i>valid</i>)	
<i>Conv. ReLU</i>	$1 \times 3 \times 192$ (<i>valid</i>)	
<i>Conv. ReLU</i>	$1 \times 1 \times 10$ (<i>valid</i>)	
<i>Global Average Pooling</i>	-	
<i>Softmax</i>	10	

APÊNDICE 2: ARQUITETURAS E HIPERPARÂMETROS DOS *AUTOENCODERS*

 TAB. 9: Arquiteturas e hiperparâmetros adotados para cada *autoencoder* do repositório *S* nos experimentos com o *dataset* MNIST.

Autoencoder	Camadas (encoder)		Camadas (decoder)		Hiperparâmetros
I	<i>Conv. Sigmoid</i>	$3 \times 3 \times 3$ (<i>same</i>)	<i>Conv. Sigmoid</i>	$3 \times 3 \times 3$ (<i>same</i>)	$reg\ strength = 10^{-10}$ $batch\ size = 256$, épocas = 50 Função de otimização: <i>Adam</i> Função de custo: MSE ruído = 0.1
	<i>Average Pooling</i>	2×2	<i>Upsampling</i>	2×2	
	<i>Conv. Sigmoid</i>	$3 \times 3 \times 3$ (<i>same</i>)	<i>Conv. Sigmoid</i>	$3 \times 3 \times 3$ (<i>same</i>)	
	-	-	<i>Conv. Sigmoid</i>	$1 \times 3 \times 3$ (<i>same</i>)	
	-	-	<i>Conv. Sigmoid</i>	$1 \times 28 \times 28$ (<i>same</i>)	
II	-	-	-	-	$reg\ strength = 10^{-10}$ $batch\ size = 256$, épocas = 50 Função de otimização: <i>Adam</i> Função de custo: MSE ruído = 0.1
	<i>Conv. Sigmoid</i>	$3 \times 3 \times 3$ (<i>same</i>)	<i>Conv. Sigmoid</i>	$3 \times 3 \times 3$ (<i>same</i>)	
	-	-	<i>Conv. Sigmoid</i>	$1 \times 3 \times 3$ (<i>same</i>)	
	-	-	<i>Conv. Sigmoid</i>	$1 \times 28 \times 28$ (<i>same</i>)	
	-	-	-	-	
III	<i>Conv. ReLU</i>	$3 \times 3 \times 3$ (<i>same</i>)	<i>Conv. ReLU</i>	$3 \times 3 \times 3$ (<i>same</i>)	$reg\ strength = 10^{-10}$ $batch\ size = 256$, épocas = 50 Função de otimização: <i>Adam</i> Função de custo: MSE ruído = 0.2
	-	-	<i>Conv. ReLU</i>	$1 \times 3 \times 3$ (<i>same</i>)	
	-	-	<i>Conv. Sigmoid</i>	$1 \times 28 \times 28$ (<i>same</i>)	
	-	-	-	-	
	-	-	-	-	
IV	<i>Conv. Sigmoid</i>	$3 \times 3 \times 5$ (<i>same</i>)	<i>Conv. Sigmoid</i>	$3 \times 3 \times 5$ (<i>same</i>)	$reg\ strength = 10^{-10}$ $batch\ size = 256$, épocas = 40 Função de otimização: <i>Adam</i> Função de custo: MSE ruído = 0.3
	<i>Max Pooling</i>	2×2	<i>Upsampling</i>	2×2	
	<i>Conv. Sigmoid</i>	$5 \times 3 \times 3$ (<i>same</i>)	<i>Conv. Sigmoid</i>	$5 \times 3 \times 3$ (<i>same</i>)	
	-	-	<i>Conv. Sigmoid</i>	$1 \times 3 \times 3$ (<i>same</i>)	
	-	-	<i>Conv. Sigmoid</i>	$1 \times 28 \times 28$ (<i>same</i>)	
V	<i>Conv. Sigmoid</i>	$3 \times 3 \times 3$ (<i>same</i>)	<i>Conv. Sigmoid</i>	$3 \times 3 \times 3$ (<i>same</i>)	$reg\ strength = 10^{-10}$ $batch\ size = 256$, épocas = 55 Função de otimização: <i>Adam</i> Função de custo: MSE ruído = 0.1
	<i>Average Pooling</i>	2×2	<i>Upsampling</i>	2×2	
	<i>Conv. Sigmoid</i>	$3 \times 3 \times 3$ (<i>same</i>)	<i>Conv. Sigmoid</i>	$3 \times 3 \times 3$ (<i>same</i>)	
	-	-	<i>Conv. Sigmoid</i>	$1 \times 3 \times 3$ (<i>same</i>)	
	-	-	<i>Conv. Sigmoid</i>	$1 \times 28 \times 28$ (<i>same</i>)	
VI	-	-	-	-	$reg\ strength = 10^{-10}$ $batch\ size = 128$, épocas = 50 Função de otimização: <i>Adam</i> Função de custo: MSE ruído = 0.2
	<i>Conv. ReLU</i>	$5 \times 3 \times 3$ (<i>same</i>)	<i>Conv. ReLU</i>	$5 \times 3 \times 3$ (<i>same</i>)	
	-	-	<i>Conv. ReLU</i>	$1 \times 3 \times 3$ (<i>same</i>)	
	-	-	<i>Conv. Sigmoid</i>	$1 \times 28 \times 28$ (<i>same</i>)	
	-	-	-	-	
VII	<i>Conv. ReLU</i>	$3 \times 3 \times 4$ (<i>same</i>)	<i>Conv. ReLU</i>	$3 \times 3 \times 4$ (<i>same</i>)	$reg\ strength = 10^{-10}$ $batch\ size = 128$, épocas = 50 Função de otimização: <i>Adam</i> Função de custo: MSE ruído = 0.1
	<i>Max Pooling</i>	2×2	<i>Upsampling</i>	2×2	
	<i>Conv. ReLU</i>	$4 \times 3 \times 3$ (<i>same</i>)	<i>Conv. ReLU</i>	$4 \times 3 \times 3$ (<i>same</i>)	
	-	-	<i>Conv. ReLU</i>	$1 \times 3 \times 3$ (<i>same</i>)	
	-	-	<i>Conv. Sigmoid</i>	$1 \times 28 \times 28$ (<i>same</i>)	
VIII	<i>Conv. ReLU</i>	$3 \times 3 \times 3$ (<i>same</i>)	<i>Conv. ReLU</i>	$3 \times 3 \times 3$ (<i>same</i>)	$reg\ strength = 10^{-10}$ $batch\ size = 256$, épocas = 35 Função de otimização: <i>Adam</i> Função de custo: MSE ruído = 0.3
	<i>Max Pooling</i>	2×2	<i>Upsampling</i>	2×2	
	<i>Conv. ReLU</i>	$3 \times 3 \times 3$ (<i>same</i>)	<i>Conv. ReLU</i>	$3 \times 3 \times 3$ (<i>same</i>)	
	-	-	<i>Conv. ReLU</i>	$1 \times 3 \times 3$ (<i>same</i>)	
	-	-	<i>Conv. Sigmoid</i>	$1 \times 28 \times 28$ (<i>same</i>)	
IX	<i>Conv. Sigmoid</i>	$3 \times 3 \times 2$ (<i>same</i>)	<i>Conv. ReLU</i>	$3 \times 3 \times 2$ (<i>same</i>)	$reg\ strength = 10^{-10}$ $batch\ size = 256$, épocas = 30 Função de otimização: <i>Adam</i> Função de custo: MSE ruído = 0.1
	<i>Max Pooling</i>	2×2	<i>Upsampling</i>	2×2	
	<i>Conv. Sigmoid</i>	$2 \times 3 \times 3$ (<i>same</i>)	<i>Conv. Sigmoid</i>	$2 \times 3 \times 3$ (<i>same</i>)	
	-	-	<i>Conv. Sigmoid</i>	$1 \times 3 \times 3$ (<i>same</i>)	
	-	-	<i>Conv. Sigmoid</i>	$1 \times 28 \times 28$ (<i>same</i>)	
X	<i>Conv. Sigmoid</i>	$5 \times 3 \times 3$ (<i>same</i>)	<i>Conv. Sigmoid</i>	$5 \times 3 \times 3$ (<i>same</i>)	$reg\ strength = 10^{-10}$ $batch\ size = 256$, épocas = 35 Função de otimização: <i>Adam</i> Função de custo: MSE ruído = 0.2
	<i>Average Pooling</i>	2×2	<i>Upsampling</i>	2×2	
	<i>Conv. Sigmoid</i>	$5 \times 3 \times 3$ (<i>same</i>)	<i>Conv. Sigmoid</i>	$5 \times 3 \times 3$ (<i>same</i>)	
	-	-	<i>Conv. Sigmoid</i>	$1 \times 3 \times 3$ (<i>same</i>)	
	-	-	<i>Conv. Sigmoid</i>	$1 \times 28 \times 28$ (<i>same</i>)	

TAB. 10: Arquiteturas e hiperparâmetros adotados para cada *autoencoder* do repositório *S* nos experimentos com o *dataset* CIFAR-10.

Autoencoder	Camadas (encoder)		Camadas (decoder)		Hiperparâmetros
I	<i>Conv. ReLU</i>	$3 \times 3 \times 16$	<i>Conv. ReLU</i>	$3 \times 3 \times 32$	épocas = 30, <i>batch size</i> = 32 <i>regularizer</i> = 10^{-10} taxa de aprendizado = 0.01 taxa de momento = 0.9, <i>nesterov</i> = true Função de otimização = SGD, ruído = 0.1 Função de custo = Entropia cruzada estocástica
	<i>Max Pooling</i>	2×2	<i>Upsampling</i>	2×2	
	<i>Conv. ReLU</i>	$3 \times 3 \times 32$	<i>Conv. ReLU</i>	$3 \times 3 \times 32$	
	<i>Max Pooling</i>	2×2	<i>Upsampling</i>	2×2	
	<i>Conv. ReLU</i>	$3 \times 3 \times 32$	<i>Conv. ReLU</i>	$3 \times 3 \times 32$	
II	-	-	<i>Conv. ReLU</i>	$3 \times 3 \times 3$	épocas = 30, <i>batch size</i> = 32 <i>regularizer</i> = 10^{-10} taxa de aprendizado = 0.01 taxa de momento = 0.9, <i>nesterov</i> = true Função de otimização = SGD, ruído = 0.1 Função de custo = Entropia cruzada estocástica
	-	-	<i>Sigmoid</i>	$3 \times 32 \times 32$	
	<i>Conv. ReLU</i>	$3 \times 3 \times 32$	<i>Conv. ReLU</i>	$3 \times 3 \times 32$	
	<i>Batch Normalization</i>	$3 \times 3 \times 32$	<i>Batch Normalization</i>	$3 \times 3 \times 32$	
	<i>Max Pooling</i>	2×2	<i>Upsampling</i>	2×2	
III	<i>Conv. ReLU</i>	$3 \times 3 \times 32$	<i>Conv. ReLU</i>	$3 \times 3 \times 32$	épocas = 30, <i>batch size</i> = 32 taxa de aprendizado = 0.01 taxa de momento = 0.9, <i>nesterov</i> = true Função de otimização = SGD, ruído = 0.1 Função de custo = Entropia cruzada estocástica
	<i>Max Pooling</i>	2×2	<i>Upsampling</i>	2×2	
	<i>Conv. ReLU</i>	$3 \times 3 \times 32$	<i>Conv. ReLU</i>	$3 \times 3 \times 32$	
	<i>Max Pooling</i>	2×2	<i>Upsampling</i>	2×2	
	<i>Conv. ReLU</i>	$3 \times 3 \times 32$	<i>Conv. ReLU</i>	$3 \times 3 \times 32$	
IV	-	-	<i>Conv. ReLU</i>	$3 \times 3 \times 3$	épocas = 30, <i>batch size</i> = 32 taxa de aprendizado = 0.01 taxa de momento = 0.9, <i>nesterov</i> = true Função de otimização = SGD, ruído = 0.1 Função de custo = Entropia cruzada estocástica
	-	-	<i>Sigmoid</i>	$3 \times 32 \times 32$	
	<i>Conv. ReLU</i>	$3 \times 3 \times 32$	<i>Conv. ReLU</i>	$3 \times 3 \times 32$	
	<i>Batch Normalization</i>	$3 \times 3 \times 32$	<i>Batch Normalization</i>	$3 \times 3 \times 32$	
	<i>Max Pooling</i>	2×2	<i>Upsampling</i>	2×2	
V	<i>Conv. ReLU</i>	$3 \times 3 \times 32$	<i>Conv. ReLU</i>	$3 \times 3 \times 32$	épocas = 30, <i>batch size</i> = 32 taxa de aprendizado = 0.01 taxa de momento = 0.9, <i>nesterov</i> = true Função de otimização = SGD, ruído = 0.1 Função de custo = Entropia cruzada estocástica
	<i>Batch Normalization</i>	$3 \times 3 \times 32$	<i>Batch Normalization</i>	$3 \times 3 \times 32$	
	<i>Average Pooling</i>	2×2	<i>Upsampling</i>	2×2	
	<i>Conv. ReLU</i>	$3 \times 3 \times 32$	<i>Conv. ReLU</i>	$3 \times 3 \times 32$	
	<i>Batch Normalization</i>	$3 \times 3 \times 32$	<i>Batch Normalization</i>	$3 \times 3 \times 32$	
VI	-	-	<i>Conv. ReLU</i>	$3 \times 3 \times 3$	épocas = 30, <i>batch size</i> = 32 <i>regularizer</i> = 10^{-10} taxa de aprendizado = 0.01 taxa de momento = 0.9, <i>nesterov</i> = true Função de otimização = SGD, ruído = 0.1 Função de custo = Entropia cruzada estocástica
	-	-	<i>Sigmoid</i>	$3 \times 32 \times 32$	
	<i>Conv. ReLU</i>	$3 \times 3 \times 8$	<i>Conv. ReLU</i>	$3 \times 3 \times 8$	
	<i>Max Pooling</i>	2×2	<i>Upsampling</i>	2×2	
	<i>Conv. ReLU</i>	$3 \times 3 \times 16$	<i>Conv. ReLU</i>	$3 \times 3 \times 16$	
VII	-	-	<i>Conv. ReLU</i>	$3 \times 3 \times 3$	épocas = 30, <i>batch size</i> = 32 <i>regularizer</i> = 10^{-10} taxa de aprendizado = 0.01 taxa de momento = 0.9, <i>nesterov</i> = true Função de otimização = SGD, ruído = 0.1 Função de custo = Entropia cruzada estocástica
	-	-	<i>Sigmoid</i>	$3 \times 32 \times 32$	
	<i>Conv. ReLU</i>	$3 \times 3 \times 16$	<i>Conv. ReLU</i>	$3 \times 3 \times 16$	
	<i>Batch Normalization</i>	$3 \times 3 \times 16$	<i>Batch Normalization</i>	$3 \times 3 \times 3$	
	<i>Average Pooling</i>	2×2	<i>Upsampling</i>	2×2	
VIII	<i>Conv. ReLU</i>	$3 \times 3 \times 16$	<i>Conv. ReLU</i>	$3 \times 3 \times 16$	épocas = 30, <i>batch size</i> = 36 taxa de aprendizado = 0.01, taxa de momento = 0.9, <i>nesterov</i> = true Função de otimização = SGD, ruído = 0.1 Função de custo = Entropia cruzada estocástica
	<i>Max Pooling</i>	2×2	<i>Upsampling</i>	2×2	
	<i>Conv. ReLU</i>	$3 \times 3 \times 24$	<i>Conv. ReLU</i>	$3 \times 3 \times 24$	
	-	-	<i>Conv. ReLU</i>	$3 \times 3 \times 3$	
IX	-	-	<i>Sigmoid</i>	$3 \times 32 \times 32$	épocas = 30, <i>batch size</i> = 36 <i>regularizer</i> = 10^{-10} taxa de aprendizado = 0.01 taxa de momento = 0.9, <i>nesterov</i> = true Função de otimização = SGD, ruído = 0.1 Função de custo = Entropia cruzada estocástica
	<i>Conv. ReLU</i>	$3 \times 3 \times 32$	<i>Conv. ReLU</i>	$3 \times 3 \times 32$	
	<i>Batch Normalization</i>	$3 \times 3 \times 32$	<i>Batch Normalization</i>	$3 \times 3 \times 32$	
	<i>Average Pooling</i>	2×2	<i>Upsampling</i>	2×2	
	<i>Conv. ReLU</i>	$3 \times 3 \times 32$	<i>Conv. ReLU</i>	$3 \times 3 \times 32$	
X	-	-	<i>Conv. ReLU</i>	$3 \times 3 \times 3$	épocas = 30, <i>batch size</i> = 36 <i>regularizer</i> = 10^{-10} taxa de aprendizado = 0.01 taxa de momento = 0.9, <i>nesterov</i> = true Função de otimização = SGD, ruído = 0.1 Função de custo = Entropia cruzada estocástica
	-	-	<i>Sigmoid</i>	$3 \times 32 \times 32$	
	<i>Conv. ReLU</i>	$3 \times 3 \times 16$	<i>Conv. ReLU</i>	$3 \times 3 \times 16$	
	<i>Batch Normalization</i>	$3 \times 3 \times 16$	<i>Batch Normalization</i>	$3 \times 3 \times 16$	
	<i>Average Pooling</i>	2×2	<i>Upsampling</i>	2×2	

APÊNDICE 3: HIPERPARÂMETROS DEFINIDOS PARA A REALIZAÇÃO DOS ATAQUES CONTRADITÓRIOS

TAB. 11: Hiperparâmetros definidos para cada algoritmo de ataque.

Ataque	<i>Dataset</i>	Hiperparâmetros
FGSM	MNIST	$\epsilon = 0,2$
	CIFAR-10	$\epsilon = 0,025$
BIM	MNIST	$\epsilon = 0.15; \alpha = 0,07; 50$ iterações
	CIFAR-10	$\epsilon = 0.025; \alpha = 0,01; 1.000$ iterações
<i>DeepFool</i>	MNIST	<i>overshoot</i> = 0,02; <i>max iter</i> = 50
	CIFAR-10	<i>overshoot</i> = 0,02; <i>max iter</i> = 50
CW	MNIST	<i>conf</i> = 0,0; <i>binary searches</i> = 1; <i>lrate</i> = 0,2; <i>initial const</i> = 10; <i>max iter</i> = 100
	CIFAR-10	<i>conf</i> = 0,0; <i>binary searches</i> = 1; <i>lrate</i> = 0,5; <i>initial const</i> = 1; <i>max iter</i> = 100