UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO DE CIÊNCIAS EXATAS, NATURAIS E DA SAÚDE - CCENS
COLEGIADO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO

Breno Batista da Silva

Análise de ataque Copycat em Redes Neurais Convolucionais com imagens sintéticas geradas à partir de Redes Generativas Adversárias

Alegre, ES

2021

Breno Batista da Silva

Análise de ataque Copycat em Redes Neurais Convolucionais com imagens sintéticas geradas à partir de Redes Generativas Adversárias

Trabalho de conclusão de curso apresentado ao Departamento de Computação do Centro de Ciências Exatas, Naturais e da Saúde da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Universidade Federal do Espírito Santo – UFES

Centro de Ciências Exatas, Naturais e da Saúde

Colegiado do Curso de Ciência da Computação

Orientador: Jacson Rodrigues Correia da Silva

Alegre, ES

2021

Breno Batista da Silva

Análise de ataque Copycat em Redes Neurais Convolucionais com imagens sintéticas geradas à partir de Redes Generativas Adversárias

Trabalho de conclusão de curso apresentado ao Departamento de Computação do Centro de Ciências Exatas, Naturais e da Saúde da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Aprovado em 14 de Maio de 2021:

Jacson R. C. Silva

Jacson Rodrigues Correia da Silva
Orientador

Juliana Pinheiro Campos Pirovani
UFES

Thiago Meireles Paixão
IFES

Alegre, ES
2021

à Amanda e Vânia

Agradecimentos

A Amanda, minha companheira, por todo o carinho, compreensão, incentivo e apoio durante toda a realização deste trabalho.

Aos meus pais, Ivan, Vânia e Taissa, por me mostrar que o estudo era o único caminho que me faria crescer, por ser um poço de amor e exemplo de pessoa batalhadora, e por ter se dedicado tanto a mim em meus primeiros anos de vida, respectivamente.

Ao meu irmão Bruno, que sempre esteve comigo em momentos bons e ruins, sempre cuidando e protegendo de mim.

A toda família e amigos pela confiança que sempre tiveram no meu potencial e pelos bons momentos de cumplicidade e integração.

Um agradecimento especial ao meu primo Vitor Hugo e ao professor Jacson Rodrigues. O primeiro, responsável por me apresentar o mundo da computação e sempre atuar como referência e mentor. O segundo, professor capaz de transcender a sala de aula e me inspirar em assuntos além da graduação, por toda paciência e tempo investido durante a orientação deste trabalho.

Por fim, gostaria de agradecer a todos os professores que me acompanharam durante a graduação e aos membros da banca, por aceitarem o convite e pelo tempo investido na leitura e avaliação desse trabalho.

"In god we trust,
All others bring data."
(W. Edwards Deming)

Resumo

Nos últimos anos houve um aumento na quantidade de sistemas inteligentes disponíveis na sociedade, através de algoritmos como as Redes Neurais Convolucionais. O custo para construção desses sistemas é alto pois envolve a necessidade de profissionais especialistas, o uso de recursos computacionais e a realização de muitas etapas, desde a aquisição de dados até a manutenção e monitoramento. Dado esse grande investimento, as empresas detentoras buscam proteger esses ativos e evitar que dados sensíveis sejam expostos ou que o próprio sistema possar ser copiado por um agente malicioso ou pela concorrência. O ataque de extração de conhecimento Copycat demonstrou a possibilidade de copiar Redes Neurais Convolucionais caixa preta, disponíveis via API, utilizando imagens relacionadas ao contexto do modelo alvo (domínio do problema) ou imagens arbitrárias sem relação com o domínio do problema. Um agente malicioso pode obter, criar ou produzir imagens do domínio do problema. Porém, como essa atividade tem um custo alto, a quantidade de imagens, geralmente, é escassa. Este trabalho analisa a viabilidade de estender o método de ataque Copycat com imagens sintéticas geradas à partir de uma Rede Generativa Adversária, treinada com uma pequena quantidade de imagens de domínio do problema obtidas pelo atacante. A análise dos resultados demonstra que a Rede Generativa Adversária é capaz de produzir novas imagens similares ao domínio do problema e que o ataque Copycat com essas imagens é viável, superando o método tradicional em alguns casos.

Palavras-chaves: Redes neurais convolucionais, Redes generativas adversárias, Ataques de privacidade, Extração de conhecimento, Copycat.

Lista de ilustrações

Figura 1 – Ilustração do Perceptron 18
Figura 2 – Gráfico da função de passo 19
Figura 3 – Principais funções de ativação não lineares 20
Figura 4 – Arquitetura da Rede Neural Artificial 21
Figura 5 – Performance das redes neurais com e sem Batch Normalization 22
Figura 6 – Arquitetura principal da Rede Neural Convolucional 23
Figura 7 – Ilustração do campo receptivo local 24
Figura 8 – O processo de convolução 25
Figura 9 – O processo de *pooling* 26
Figura 10 – Topologia da VGG16 e da VGG19 28
Figura 11 – Visão geral da Rede Generativa Adversária 29
Figura 12 – O processo de treino da Rede Generativa Adversária 30
Figura 13 – Topologia da Rede Generativa Adversária 31
Figura 14 – Ilustração do problema de *mode collapse* 31
Figura 15 – Correlação do FID com diferentes tipos e níveis de pertubação 33
Figura 16 – Arquiteturas propostas para Redes Generativas Adversárias 34
Figura 17 – Visão geral do ambiente em um ataque de privacidade 35
Figura 18 – Exemplo de imagens do CIFAR-10 por classe 38
Figura 19 – Exemplo de imagens do MNIST por classe 38
Figura 20 – Exemplo de imagens do SVHN por classe 39
Figura 21 – Visão geral do ataque Copycat 42
Figura 22 – Visão geral da metodologia proposta 45
Figura 23 – Desempenho durante o treinamento das Redes Generativas Adversárias 55
Figura 24 – Imagens sintéticas produzidas pela RGA-DP 56
Figura 25 – Imagens sintéticas produzidas pela RGA-NDP 57

Lista de tabelas

Tabela 1 – Detalhes dos problemas investigados 48
Tabela 2 – Conjuntos de imagens do Domínio do Problema 50
Tabela 3 – Resultado dos modelos de *baseline* 57
Tabela 4 – Resultado do ataques Copycat com imagens sintéticas de DP 58
Tabela 5 – Resultado do ataques Copycat com imagens sintéticas de NDP 59
Tabela 6 – Visão geral dos principais resultados 59

Lista de abreviaturas e siglas

API	Application Programming Interface
BN	Batch Normalization
CDA	Conjunto de Dados de Ataque
CDM	Classificação de Dígitos Manuscritos
CDO	Conjunto de Dados Original
CDV	Conjunto de Dados de Validação
CNCVR	Classificação de Número da Casa Visto da Rua
COG	Classificação de Objetos Gerais
DP	Domínio do Problema
FID	Fréchet Inception Distance
IA	Inteligência Artificial
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
IS	Inception Score
LGPD	Lei Geral de Proteção a Dados
MS	Mode Seeking
NDP	Não Domínio do Problema
RGA	Rede Generativa Adversária
RNA	Rede Neural Artificial
RNC	Rede Neural Convolucional
RNP	Rede Neural Profunda
RO	Rótulos Originais
RR	Rótulos Roubados
SVM	Support Vector Machines
TA	Taxa de Aprendizado

Sumário

1 INTRODUÇÃO . . . 11
1.1 Objetivos . . . 14
1.1.1 Objetivo geral . . . 14
1.1.2 Objetivos específicos . . . 14
1.2 Organização deste trabalho . . . 14

2 REFERENCIAL TEÓRICO . . . 16
2.1 Aprendizado de Máquina . . . 16
2.1.1 Redes Neurais Artificiais . . . 18
2.1.2 Redes Neurais Convolucionais . . . 22
2.1.3 Redes Generativas Adversárias . . . 29
2.2 Ataques de Privacidade . . . 34
2.3 Conjuntos de Dados de Domínio Público . . . 37

3 TRABALHOS RELACIONADOS . . . 40

4 METODOLOGIA . . . 45
4.1 Modelos Alvo para Realização dos Ataques . . . 47
4.2 Ataque Copycat para Comparação dos Resultados . . . 48
4.3 Treinamento da RGA para Gerar Imagens Sintéticas . . . 49
4.4 Ataque Copycat com Imagens Geradas pela RGA . . . 51
4.5 Comparação dos Resultados . . . 52
4.6 Implementação e Configurações . . . 52
4.7 Recursos Computacionais . . . 53

5 RESULTADOS . . . 54
5.1 Análise do Treinamento da RGA e das Imagens Sintéticas . . . 54
5.2 Resultados dos Modelos Gerados . . . 57

6 CONCLUSÃO . . . 61

REFERÊNCIAS . . . 62

1 Introdução

O aprendizado de máquina é o campo de pesquisa da Inteligência Artificial (IA) que estuda o desenvolvimento e a aplicação de algoritmos capazes de reconhecer padrões, relacionamentos e estruturas em dados sem serem explicitamente programados (SAMUEL, 1959). O campo de aprendizado de máquina possui diversos estudos que puderam solucionar problemas atuais na computação e na sociedade. Dentre as possíveis soluções de algoritmos existentes, descritos no Capítulo 2, a Rede Neural Convolucional (RNC), é capaz de resolver problemas relacionados ao processamento de dados como imagens, texto, áudio e vídeo, apresentando os melhores resultados da literatura (GOODFELLOW et al., 2014). Com seu crescente uso em sistemas inteligentes, as RNCs estão cada vez mais presentes no dia a dia da sociedade, em assistentes pessoais, como a Alexa da Amazon, ou mesmo em celulares em geral. Outras aplicações mais específicas são, por exemplo, sistemas capazes de identificar e analisar sentimentos (DANG; MORENO-GARCÍA; PRIETA, 2020), reconhecer placas de veículos em tempo real (YANG; WANG, 2019) e identificar câncer de mama em Raio-X (RASHED; SEOUD, 2019).

Além da crescente quantidade de pesquisas nesse campo, também há um crescimento no número de pequenas empresas cujo o produto principal está relacionado a sistemas inteligentes com aprendizado de máquina. O financiamento desses produtos é fornecido por empresas de Capital de Risco (HAGENDORFF, 2020), pois o custo de construção de sistemas de aprendizado de máquina é alto, principalmente devido à contratação de especialistas no assunto e ao uso de recursos computacionais caros. Tais sistemas inteligentes costumam utilizar um conjunto de dados previamente construído, denominado de Conjunto de Dados Original (CDO). Esse conjunto de dados possui um custo alto para produção pois sua importância está diretamente relacionada à capacidade de interpretação e de generalização do modelo.

A presença de vieses ou correlações espúrias no CDO pode gerar impacto sociais, como o sistema inteligente que considerou a pele escura como menos atrativa, ao julgar um concurso de beleza (CATON; HAAS, 2020). Outra preocupação relacionada à privacidade dos dados utilizados para gerar um sistema de aprendizado de máquina, ou mesmo da cópia do próprio sistema, deve-se às informações utilizadas serem dados sensíveis de pessoas ou de empresas. Devido a tais fatores, existe um grande interesse em mecanismos de segurança para proteger o vazamento dos dados, os detalhes da implementação e os parâmetros desses modelos para empresas concorrentes. Até mesmo leis de proteção de dados tem sido discutidas ao redor do mundo, inclusive aqui no Brasil, como a Lei Geral de Proteção a Dados (LGPD).

Além disso, o uso desses sistemas inteligentes na sociedade tem incentivado os estudos em ataques de privacidade em aprendizado de máquina. Tais estudos objetivam explorar e também obter informações sobre o CDO, sobre a arquitetura utilizada no algoritmo ou sobre os parâmetros do sistema, os quais representam seu conhecimento (RIGAKI; GARCIA, 2020). Os ataques existentes são classificados em caixa branca quando o atacante possui conhecimento prévio da arquitetura ou hiper-parâmetros que foram utilizados na construção do modelo, ou em caixa preta, quando o atacante possui acesso somente a Application Programming Interface (API) do modelo alvo, sendo somente possível enviar um conjunto de amostras e obter, como resposta, as predições fornecidas pelo modelo (RIGAKI; GARCIA, 2020).

Um dos tipos de ataques estudados na literatura é a extração de modelos, que tem o objetivo de extrair informações do modelo alvo para construir um modelo substituto que produz respostas equivalentes ou semi-equivalentes ao original (RIGAKI; GARCIA, 2020). Nesse escopo de pesquisa, o atacante precisa gerar um Conjunto de Dados de Ataque (CDA) com pares de dados formados por: imagens fornecidas pelo atacante e predições fornecidas pelo modelo alvo. As imagens podem ser naturais aleatórias (CORREIA-SILVA et al., 2018; CORREIA-SILVA et al., 2021)), imagens do domínio do problema (TRAMÈR et al., 2016; SHI; SAGDUYU; GRUSHIN, 2017; CORREIA-SILVA et al., 2018; CORREIA-SILVA et al., 2021) e imagens sintéticas (CHEN et al., 2019; YUAN et al., 2020). Como predições, podem ser utilizados os rótulos (*hard labels*) ou, quando fornecidas, as probabilidades (*soft labels*) do modelo alvo.

Dentre os diferentes estudos de extração de modelos, a Copycat (CORREIA-SILVA et al., 2018; CORREIA-SILVA et al., 2021) é um dos métodos que mais se adéqua às restrições que um atacante encontrará em um cenário real. Nesse método o modelo alvo é uma caixa preta disponível através de uma API, que retorna somente os rótulos para cada imagem de entrada e nenhum conhecimento prévio do CDO, arquitetura ou parâmetros do modelo são conhecidos pelo atacante. A Copycat é capaz de copiar modelos construídos para diferentes tarefas de classificação, utilizando somente imagens naturais aleatórias (imagens do ImageNet (DENG et al., 2009)), conhecidas como Não Domínio do Problema (NDP). Imagens NDP estão disponíveis em abundância na Internet ou em conjuntos de dados públicos. Porém, não há garantia que o modelo alvo produza rótulos para todas as categorias desejadas de extração do modelo alvo, sendo necessário que o atacante utilize uma diversidade de imagens para compor o CDA e, assim, explorar com sucesso o espaço de classificação do modelo alvo (CORREIA-SILVA et al., 2021).

Outro destaque em ataques semelhantes é a utilização de imagens sintéticas produzidas por algoritmos generativos (CHEN et al., 2019; YUAN et al., 2020). A abordagem de uso de tais imagens sintéticas demonstrou ser capaz de mapear o espaço de classificação dos modelos alvo, permitindo que o modelo substituto alcance acurácias similares aos

modelos alvo. Os algoritmos generativos possuem o objetivo de produzir novas imagens a partir de um conjunto de imagens que seguem uma distribuição desconhecida.

Dentre os algoritmos generativos existentes, descritos no Capítulo 2, o que possui mais destaque atualmente é a Rede Generativa Adversária (RGA). Proposta por Goodfellow et al. (2014), a RGA é o estado-da-arte em algoritmos generativos e seu treinamento é baseado em uma competição entre duas redes neurais. A primeira, conhecida como Gerador, possui a tarefa de produzir novas imagens com o objetivo de "enganar" a segunda, conhecida como Discriminador, cujo objetivo é classificar as imagens como falsas (i.e., produzidas pelo Gerador) ou como verdadeiras (i.e., imagens fornecidas para o sistema).

Tal rede foi utilizada no trabalho de Chen et al. (2019), onde o modelo alvo desempenhou o papel de Discriminador. O Gerador foi treinado para produzir imagens sintéticas à partir de ruídos aleatórios, sendo então utilizadas para copiar o modelo alvo. Embora seja um dos primeiros trabalhos a demonstrar a possibilidade de efetuar um ataque de extração sem nenhum conjunto de imagens prévio, o método requer que uma larga quantidade de requisições sejam feitas à API do modelo a cada iteração da RGA. Isso pode tornar o método inviável para um ataque de extração em um cenário real, em que cada requisição à API do modelo alvo possui um custo associado, ou que existam métodos de defesa capazes de identificar uma quantidade grande de consultas com imagens de fora do Domínio do Problema (DP).

Na literatura também existem trabalhos que realizaram ataques utilizando imagens do DP, apresentando boa performance e capacidade de cópia (TRAMÈR et al., 2016; SHI; SAGDUYU; GRUSHIN, 2017; CORREIA-SILVA et al., 2018), possivelmente pelo fato de melhor explorar o espaço de classificação do modelo alvo. Mas em um cenário real, nem sempre essas imagens estão disponíveis ao atacante e o custo para sua obtenção é alto, uma vez que é necessário obtê-las, criá-las, ou coletá-las. Além disso, também é de custo alto sua anotação por um serviço da Internet ou manualmente pelo atacante. Adicionalmente, mesmo que o atacante consiga gerar esse conjunto de dados, essas imagens podem ter rótulos diferentes dos que seriam fornecidos pelo modelo alvo.

Para contornar esse obstáculo, o atacante pode obter um conjunto pequeno de imagens relacionadas ao DP na Internet e então anotá-las utilizando os rótulos do modelo alvo. Após isso, evitando utilizar o modelo alvo como Discriminador, assim como o método de Chen et al. (2019), o atacante pode utilizar esses pares de imagem e rótulo para treinar uma RGA e, então, gerar novas imagens sintéticas mais próximas ao espaço dimensional do DP desejado. Da mesma forma, essas imagens sintéticas podem ser anotadas pelo modelo alvo, permitindo obter mais informações do próprio modelo atacado em relação ao seu espaço de classificação, formando um conjunto de dados local com informações possivelmente suficientes para que o atacante possa alcançar uma cópia equivalente ou semi-equivalente ao modelo alvo. Baseando-se nessas suposições, esse trabalho estende o

método apresentado em (CORREIA-SILVA et al., 2018), mas tem como hipótese que é possível atacar modelos de RNCs em caixa preta com imagens sintéticas geradas por RGA ao invés de utilizar imagens de NDP obtidas na Internet. Para verificar a acurácia desse ataque, propõe-se a comparação do resultado desse modelo com o resultado do método apresentado em (CORREIA-SILVA et al., 2018), utilizando um mesmo conjunto de testes entre tais modelos.

No desenvolvimento do trabalho, foram utilizados conjuntos públicos de imagens obtidos na Internet. Assim, formaram-se os conjuntos de treinamento do modelo alvo, os conjuntos de domínio do problema para o ataque e os conjuntos de testes para as comparações. Com o método proposto, foi possível alcançar uma acurácia superior a 85% para todos os problemas investigados, obtendo desempenho similar ou superior ao modelo apresentado por Correia-Silva et al. (2018).

1.1 Objetivos

1.1.1 Objetivo geral

Analisar a possibilidade de ataques do tipo Copycat em Redes Neurais Convolucionais com imagens sintéticas geradas a partir de Redes Generativas Adversárias.

1.1.2 Objetivos específicos

1. Gerar modelos alvo para realizar os ataques.
2. Gerar um ataque Copycat para a comparação de resultados.
3. Treinar as RGAs para gerar novas imagens de ataque.
4. Efetuar o ataque Copycat com as imagens geradas pelas RGAs.

1.2 Organização deste trabalho

O restante do texto está estruturado da seguinte forma:

- O Capítulo 2 apresenta os assuntos e aspectos relativos ao conteúdo teórico relevante para o trabalho;
- O Capítulo 3 apresenta os trabalhos desenvolvidos que estão relacionados com a proposta deste trabalho e suas principais diferenças;
- O Capítulo 4 apresenta a metodologia empregada na realização dos objetivos e os recursos computacionais utilizados;

- O Capítulo 5 apresenta os resultados obtidos nos experimentos, análise dos resultados e a comparação contra os resultados da literatura;

- O Capítulo 6 apresenta as considerações finais do trabalho, como contribuições, limitações, objetivos, resultados alcançados e melhorias para trabalhos futuros.

2 Referencial Teórico

Neste capítulo são apresentados os conceitos teóricos necessários para realização e compreensão deste trabalho. Inicialmente, são descritas as definições e os tipos de aprendizado de máquina, apresentando com mais detalhes alguns dos algoritmos presentes no aprendizado de máquina profundo. Em seguida são explicados o que são ataques de privacidade, como é a configuração de um ataque e quais os tipos conhecidos, em especial os ataques de extração de modelos. Por fim, os conjuntos de dados comuns na literatura, principalmente os relacionados aos problemas investigados neste trabalho, são apresentados.

2.1 Aprendizado de Máquina

Samuel (1959) definiu o aprendizado de máquina como o campo de estudo que dá aos computadores a capacidade de aprender sem ser explicitamente programado. Essa capacidade é necessária pois existem muitos problemas que não podem ser resolvidos por métodos clássicos de programação, uma vez que não é possível modelar esses problemas matematicamente ou algoritmicamente (PRATI, 2006). Era desconhecido, por exemplo, como escrever um programa de computador que reconhecesse números escritos à mão, mas já existia uma grande quantidade de imagens de números escritos à mão disponíveis na Internet (PRATI, 2006). Assim, a aprendizagem de máquina estuda a construção de algoritmos que permitem que os computadores automatizem a construção e programação de modelos baseados em dados por meio de uma descoberta sistemática de padrões estatisticamente significativos nos dados disponíveis (CHOWDHURY; APON; DEY, 2017). Estes dados podem vir da natureza, serem feitos à mão por humanos ou gerados por outro algoritmo (BURKOV, 2019). Os algoritmos de aprendizado de máquina podem ser divididos em: supervisionado, não supervisionado e por reforço (RUSSELL; NORVIG, 2002).

No aprendizado supervisionado, um conjunto de n amostras forma o conjunto de dados do problema O, sendo $O = \{(x_i, y_i)\}_{i=1}^{n}$. Assim, sabe-se qual é a classe verdadeira ou aproximadamente verdadeira que cada observação pertence (RUSSELL; NORVIG, 2002). Cada observação x_i é um vetor, chamado de vetor de características $\mathbb{R}^D$, em que cada dimensão $j = 1, ..., D$ contém um valor que a descreve de alguma forma. Este valor é chamado de *feature* e é denotado por $x_i^{(j)}$ (BURKOV, 2019). Para todas as observações $x \in O$, a *feature* na posição j deve sempre representar a mesma característica. Por exemplo, se a feature $x_i^{(2)}$ contém o peso em quilogramas da pessoa x_i, então $x_k^{(2)}$ também contém o peso em quilogramas da pessoa $x_k, k = 1, ..., n$ (BURKOV, 2019). O valor y_i pode ser

um número real ou um elemento que pertence a um conjunto finito de classes $\{1, 2, ..., C\}$. Quando o valor y_i pertence a um conjunto de classes, deve-se utilizar algoritmos de classificação, e quando é um número real, deve-se utilizar algoritmos de regressão. Além disso, y_i também pode ser uma estrutura complexa como vetores, matrizes, árvores ou grafos (BURKOV, 2019).

O objetivo de um algoritmo de aprendizado supervisionado é usar o conjunto de dados O para construir um modelo h, chamado de hipótese, que se aproxime ao máximo da função original desconhecida f. Esse processo é chamado de treinamento do modelo (RUSSELL; NORVIG, 2002). Após treinado, o modelo h deve ser capaz de, ao receber como entrada um novo vetor de *features* $\bar{x}$, que não pertence ao conjunto de dados utilizado em seu treinamento (ou seja, $\bar{x} \notin O$), gerar uma saída $h(\bar{x}) = \hat{y}$ que seja a mais próxima possível da classe desejada ou do valor $y = f(x)$ original (BURKOV, 2019).

Já no aprendizado não supervisionado, o conjunto de dados não é rotulado $O = \{x_i\}_{i=1}^{N}$ e os algoritmos tem o objetivo de construir um modelo capaz de aprender o espaço hiperdimensional do conjunto de dados de entrada, para executar alguma das seguintes tarefas: agrupamento, redução de dimensionalidade ou detecção de anomalias. No agrupamento, o modelo recebe um vetor de *features* $\bar{x}$ como entrada e retorna a qual grupo $\bar{x}$ pertence. Em redução de dimensionalidade, a entrada é o vetor de *features* $\bar{x}$ e o retorno é um novo vetor $\bar{k}$ que possui uma quantidade de features menor que $\bar{x}$. Por fim, na detecção de anomalias, o retorno é um número real que indica o quanto $\bar{x}$ é diferente dos vetores do conjunto de dados O original (BURKOV, 2019).

Entre o aprendizado supervisionado e o não supervisionado existe um meio termo chamado de aprendizado semisupervisionado. Nesse tipo, o conjunto de dados possui observações com rótulos e observações sem rótulos, mantendo o mesmo objetivo do aprendizado supervisionado (PRATI, 2006). As observações sem rótulo trazem mais informações sobre a distribuição original do evento que o modelo busca aproximar e por isso pode ser útil (BURKOV, 2019).

E no aprendizado por reforço, o sistema desenvolvido possui interações com o ambiente que o cerca e aprende uma política ótima, ou subótima, de ação por experimentação direta com o meio. O agente é recompensado ou penalizado, dependendo da ação executada. Seu objetivo é desenvolver uma política que maximize a quantidade de recompensa recebida ao longo de sua execução (PRATI, 2006).

Além de diferentes tipos, o aprendizado de máquina possui diferentes paradigmas, como: simbólico, estatístico, baseado em protótipo, conexionista e genético (PRATI, 2006). O escopo desse trabalho é o paradigma conexionista, englobando Redes Neurais Artificiais, Redes Neurais Convolucionais e Redes Neurais Adversariais. O decorrer dessa seção busca melhor explicar esses métodos.

2.1.1 Redes Neurais Artificiais

As Redes Neurais Artificiais (RNAs) são construções matemáticas, relativamente simples, e seus estudos foram iniciados com a pesquisa de McCulloch e Pitts (1943), inspiradas em modelos biológicos do sistema nervoso, formado pelos neurônios. Os neurônios são unidades funcionais compostos de dendritos e axônios, que recebem e mandam informações eletroquímicas de e para outros neurônios, ocorrendo a troca de informações através de suas conexões: as sinapses. Dessa forma, as RNAs são redes que ligam pequenas células de processamento de dados, sendo cada uma chamada de neurônio artificial, que assim como o neurônio biológico, recebe, processa e envia informação (RUSSELL; NORVIG, 2002).

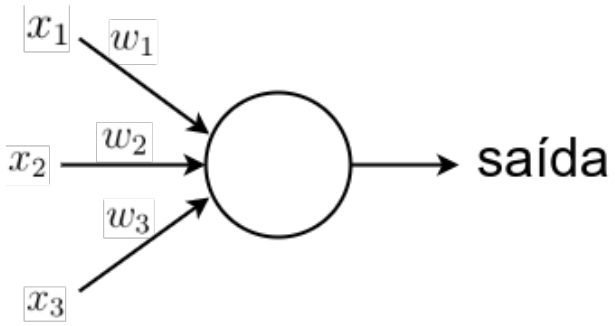

Figura 1 – Ilustração do Perceptron. O conjunto de entradas x_1, x_2, x_3 é multiplicado pelos pesos w_1, w_2, w_3 de cada respectiva aresta e somado para produzir a saída. Imagem adaptada de Nielsen (2015)

Inspirado no trabalho de McCulloch e Pitts (1943), Rosenblatt (1961) apresentou o Perceptron, que é a base dos neurônios artificiais utilizados atualmente. Um Perceptron recebe um conjunto de entradas $(x_1, x_2, x_3, ..., x_n)$, que após serem multiplicadas pelos valores das arestas conectadas, definidos como pesos $(w_1, w_2, w_3, ..., w_n)$, produz uma única saída, conforme a Figura 1 (NIELSEN, 2015). Os pesos $(w_1, w_2, w_3, ..., w_n)$ representam o conhecimento (NIELSEN, 2015) e a saída do Perceptron é binária e foi definida por Rosenblatt (1961) conforme a Equação 2.1.

$$saída = \begin{cases} 0 & if \quad \sum_j w_j x_j \ \leq \ limiar \\ 1 & if \quad \sum_j w_j x_j \ > limiar \end{cases} \tag{2.1}$$

A saída dos neurônios pode ser definida conforme a Equação 2.2, onde F é chamado de função de ativação, definindo o formato da saída Y do neurônio. Essa função de ativação F, é uma transformação linear ou não linear que pode variar de acordo com a função escolhida (GOODFELLOW et al., 2016).

$$Y = F(\Sigma(peso * entrada)) \tag{2.2}$$

No caso do Perceptron, F é a função de passo, que é uma transformação linear,

ilustrada na Figura 2, e o resultado de F é quem define se a informação do Perceptron é relevante e deve ser passada em frente ($F = 1$) ou não ($F = 0$) (NIELSEN, 2015). Devido a essa saída binária dos Perceptrons, Minsky e Papert (2017) demonstraram que uma rede de Perceptrons é incapaz de representar e resolver problemas que não sejam linearmente separáveis. Outro problema também provocado pela função de passo é que durante o aprendizado de uma rede de Perceptrons, uma pequena atualização nos pesos pode resultar em uma mudança drástica na saída, de 0 para 1, ou 1 para 0, impactando todo o restante da rede (NIELSEN, 2015).

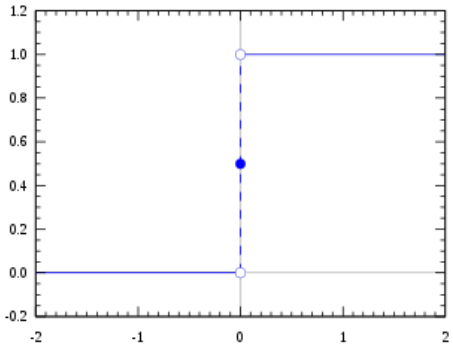

Figura 2 – Gráfico da função de passo. Essa função retorna 1 para valores maiores ou igual a 0 da entrada e 0 para valores negativos da entrada. Imagem adaptada de Szandała (2020)

Por isso, a função de passo foi substituída pela função Sigmoid, definida e ilustrada na Figura 3, transformando a saída dos neurônios de binária para contínua, com valores entre 0 e 1. Isso possibilitou que as redes sejam capazes de resolver problemas linearmente não separáveis, e que pequenas atualizações nos pesos ocorram sem gerar mudanças drásticas na saída dos neurônios (NIELSEN, 2015). Além da função de passo e da Sigmoid, diversas outras funções de ativação existem, com as principais representadas na Figura 3.

Além das funções de ativação, outro recurso utilizado na construção das RNAs é o Dropout, apresentado por Srivastava et al. (2014). O Dropout desliga de forma aleatória alguns neurônios de cada camada, dado uma probabilidade p previamente escolhida, em cada etapa do treinamento. Como alguns neurônios estão desligados, a informação é forçada a passar por outros neurônios, evitando que alguns neurônio dominem o conhecimento da rede. Isso introduz uma regularização na rede e aumenta a capacidade de generalização da rede (KHAN et al., 2020).

Existem diversas arquiteturas de RNAs, como a Rede Neural Recorrente (ELMAN, 1990) e a Rede Neural sem Peso (LUDERMIR; SOUTO; OLIVEIRA, 2008). Porém, a mais utilizada é a Rede Neural Feedforward, que recebe esse nome porque a informação flui através das conexões das camadas da esquerda para direita, sem nenhuma conexão de retorno (GOODFELLOW et al., 2016). Uma Rede Neural Feedforward é formada por camadas de neurônios, conectadas em sequência, conforme a Figura 4, onde cada neurônio

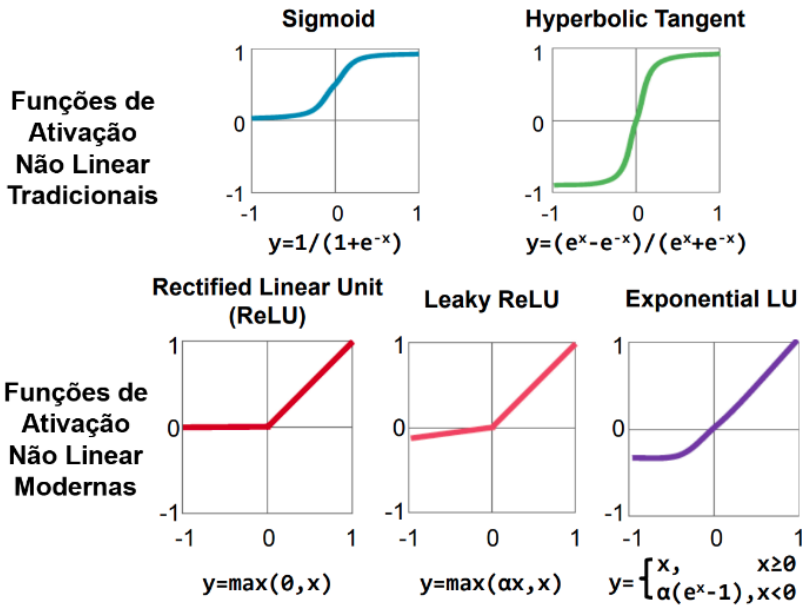

Figura 3 – Principais funções de ativação não lineares. Essas funções podem ser divididas em tradicionais e modernas, sendo elas: (a) A Sigmoid, que transforma os valores de entrada em valores contínuos entre 0 a 1, o que permite que a função seja utilizada como uma função probabilística; (b) A Hyperbolic Tangent, que funciona de forma semelhante a Sigmoid, mas entrega valores contínuos entre -1 a 1; (c) A ReLU (NAIR; HINTON, 2010), que mantém valores os positivos inalterados e transforma todos os valores negativos em 0; (d) A Leaky ReLU (MAAS; HANNUN; NG, 2013), que assim como a ReLU, mantém os valores positivos inalterados e transforma os valores negativos em uma reta com uma inclinação α, o que previne que o gradiente desapareça entre as camadas durante o *backpropagation*; (e) Por último, a Exponential LU (CLEVERT; UNTERTHINER; HOCHREITER, 2015) mantém valores positivos e transforma os negativos através de uma função exponencial. Imagem adaptada de Sze et al. (2017)

dessas camadas recebem informações da camada anterior e, após executar o processamento e a função de ativação, emitem o resultado como saída para os neurônios da próxima camada. A única exceção são os neurônios da camada de entrada, que recebem o vetor de *features* das amostras como entrada (GOODFELLOW et al., 2016). Quando todos os neurônios de cada camada estão ligados a todos os neurônios da camada anterior e da próxima, assim como o exemplo da Figura 4, a rede é chamada de totalmente conectada.

O treinamento da RNA consiste na alteração dos parâmetros w_{ij} (peso do neurônio i para o j), com o objetivo de minimizar a diferença entre o valor de saída e o valor esperado (HAYKIN et al., 2009). Para alterar esses parâmetros, usa-se a derivada da função $f_l(\cdot)$, aplicada em cada neurônio de cada camada da rede, onde $l = 1, \ldots, L$, com L sendo o número total de camadas da rede. As derivadas das funções formam seu gradiente, indicando a direção onde o erro da rede (diferença entre o valor de saída e do valor esperado

da rede) aumenta. Assim, utiliza-se o negativo do gradiente para corrigir os parâmetros da rede, indo da última camada até a primeira através do algoritmo *backpropagation* (RUMELHART; HINTON; WILLIAMS, 1985). Essas correções são ponderadas por uma Taxa de Aprendizado (TA), evitando que mudanças muito grandes sejam feitas e atrapalhem a convergência da rede (GOODFELLOW et al., 2016). Dessa forma, ao minimizar o erro da rede, um melhor conjunto de parâmetros pode ser encontrado a cada iteração do algoritmo (LECUN et al., 1989).

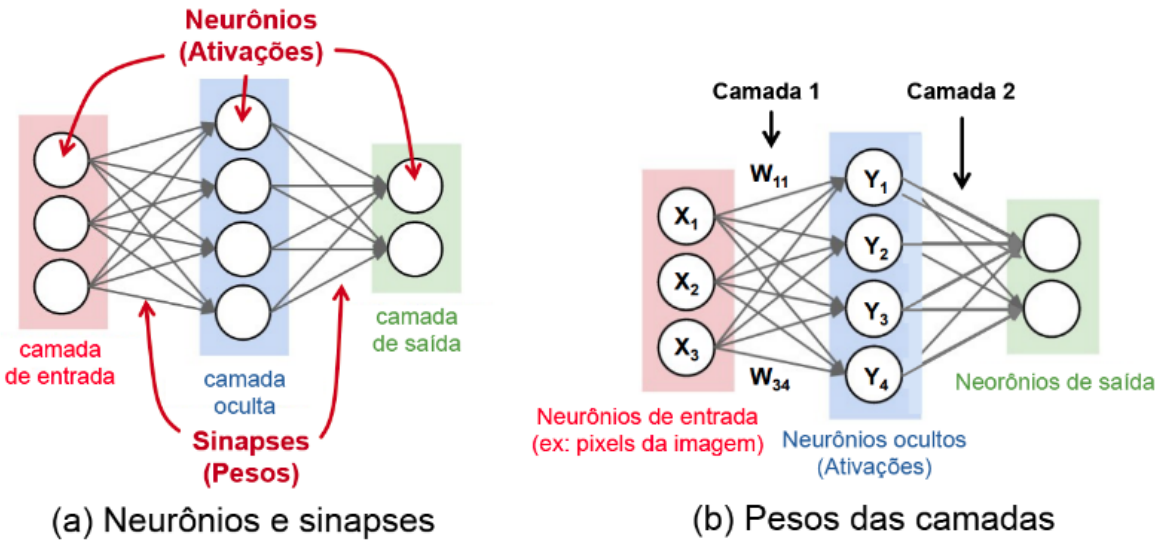

Figura 4 – Arquitetura da Rede Neural Artificial. (a) Representa os neurônios, que são responsáveis pelo processamento das informações, através das funções de ativação, distribuídos entre as camadas de entrada (zona vermelha), camadas ocultas (zona azul) e camadas de saída (zona verde), e as sinapses, que são as conexões entre os neurônios. (b) Representa os pesos w_{ij} das camadas, onde i é o número do neurônio origem da informação e j é o número do neurônio destino. Esses pesos são multiplicados por x_i e somados, para produzir y_j. Imagem adaptada de Sze et al. (2017)

Com a evolução dos recursos computacionais, RNAs com mais camadas ocultas foram construídas, buscando aumentar a precisão e generalização (UTGOFF; STRACUZZI, 2002). As Redes Neurais Profundas (RNPs) são RNAs que utilizam um maior número de camadas ocultas e pertencem ao grupo de algoritmos de aprendizado profundo (BURKOV, 2019).

Algoritmos de aprendizado raso, que são a grande maioria dos algoritmos de aprendizado de máquina, aprendem os parâmetros do modelo diretamente das variáveis de entrada através de observações sobre o conjunto de treino. Já os algoritmos de aprendizado profundo aprendem os parâmetros não diretamente das variáveis de entrada do treino, mas das saídas das camadas anteriores (BURKOV, 2019) e isso permite que esses modelos aprendam representações de dados com vários níveis de abstração. Esses métodos melhora-

ram drasticamente o estado da arte em reconhecimento de fala, reconhecimento de objeto visual, detecção de objeto e muitos outros domínios (LECUN; BENGIO; HINTON, 2015).

Outra característica da RNP é a utilização de Batch Normalization (BN), apresentado por Ioffe e Szegedy (2015), sendo um tipo especial de camada que executa uma normalização dos valores de entrada das camadas internas para cada lote de treinamento. Essa normalização visa diminuir mudanças grandes da covariância interna, o que permite a utilização de taxas de aprendizados maiores e também atua como um regularizador, eliminando a necessidade do Dropout em alguns casos (KHAN et al., 2020). A Figura 5 apresenta os resultados encontrados por Bjorck et al. (2018) utilizando redes com e sem BN.

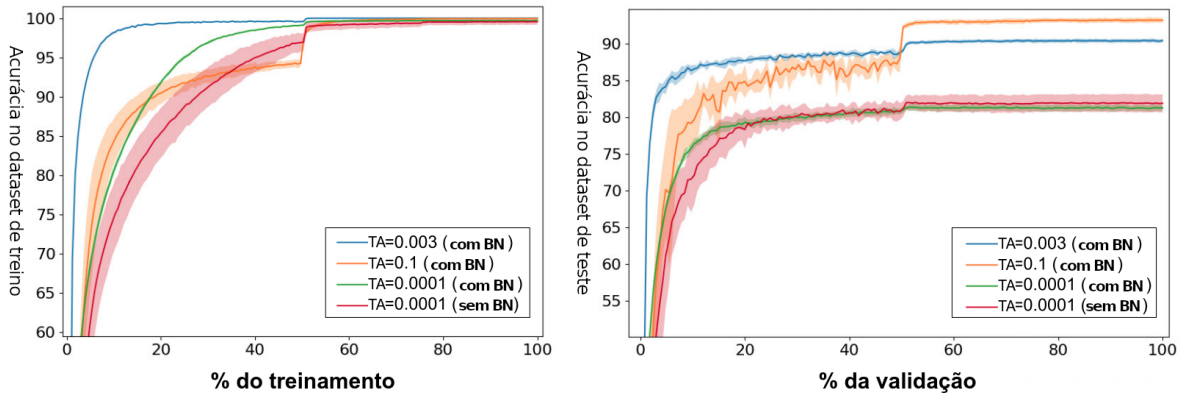

Figura 5 – Performance das redes neurais com e sem Batch Normalization. A imagem da esquerda representa a evolução da performance das redes (eixo y), durante as épocas de treinamento (eixo x). A imagem da direita representa a mesma informação, mas para as épocas de validação. A linha vermelha representa uma rede treinada sem BN e TA igual a 0.0001, enquanto a linha verde representa uma rede treinada com BN e TA também igual a 0.0001. A rede verde, apesar de convergir para a mesma performance que a rede vermelha, converge de forma mais rápida. A linha azul representa uma rede treinada com BN e TA igual a 0.003 e a linha amarela representa uma rede treinada com BN e TA igual a 0.1. Imagem adaptada de Bjorck et al. (2018)

Dentre as RNPs, destacam-se as Redes Neurais Convolucionais (RNCs), sendo atualmente a base para muitas aplicações modernas de aprendizado de máquina. Elas trouxeram avanços no processamento de imagens, de vídeos e de fala, com aplicações em carros autônomos, detecção de câncer e em jogos complexos (SZE et al., 2017).

2.1.2 Redes Neurais Convolucionais

LeCun et al. (1990) apresentou uma RNA que utilizava camadas de convolução, um tipo especializado de operação matemática, para reconhecer dígitos manuscritos. A chamada LeNet é reconhecida como a primeira Rede Neural Convolucional e foi aprimorada ao longo dos anos (LECUN et al., 1989; LECUN et al., 1990; LECUN et al., 1998).

A convolução permite que redes profundas aprendam funções em dados como imagens, vídeo e texto. Matematicamente, as redes convolucionais fornecem cálculos para explorar efetivamente a estrutura local dos dados. Por exemplo, uma imagem é representada como uma matriz bidimensional de *pixels* e partes desta imagem que estão próximas umas das outras na matriz de *pixels* tendem a variar juntas. Desta forma, a RNC aprende a explorar essa estrutura de covariância natural para aprender de forma efetiva e extrair o melhor conjunto de características das imagens (RAMSUNDAR; ZADEH, 2018).

O treinamento de uma RNC é semelhante com o de uma RNA e, de forma geral, consiste em ajustar os parâmetros (pesos) internos da rede, de forma otimizar uma função objetivo. Esses parâmetros também são otimizados de acordo com o gradiente descendente e o algoritmo de *backpropagation* (LECUN; BENGIO; HINTON, 2015).

As RNCs evoluíram após o trabalho inicial de LeCun et al. (1990). Impulsionados pelo aumento de poder computacional, novas arquiteturas, tipos de camadas e operações matemáticas foram estudadas e apresentadas pela comunidade científica (GOODFELLOW et al., 2016). Atualmente, a maioria das arquiteturas de RNC apresentam três camadas principais, conforme a Figura 6, sendo elas: (a) camada convolucional, (b) a camada de *pooling* e (c) a camada totalmente conectada (RAWAT; WANG, 2017).

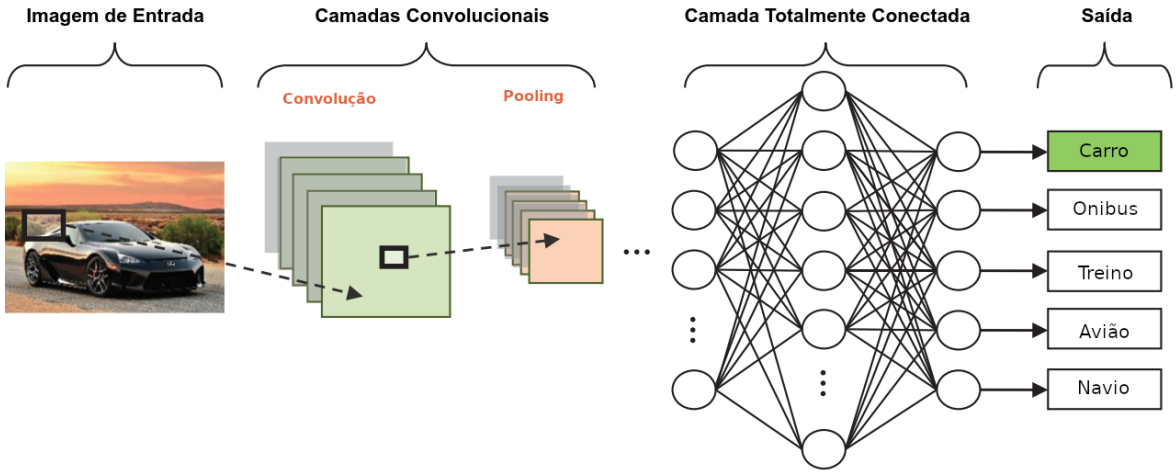

Figura 6 – Arquitetura principal da Rede Neural Convolucional. A imagem de entrada é processada através de camadas convolucionais aplicadas em sequência e depois por uma rede totalmente conectada, que produz como saída os valores ou rótulos relacionados a imagem. Imagem adaptada de Rawat e Wang (2017)

A camada convolucional utiliza o conceito de campo receptivo local dos neurônios, que é a parte da percepção sensorial do corpo humano, que afeta o disparo do neurônio. Esse campo de visão pode corresponder a um pedaço de pele ou a um segmento do campo visual de uma pessoa (RAMSUNDAR; ZADEH, 2018). Aplicado às imagens, cada campo receptivo corresponde a uma porção de *pixels*, conforme apresenta a Figura 7.

O campo receptivo local se desloca ao longo dos eixos x e y da matriz da imagem I

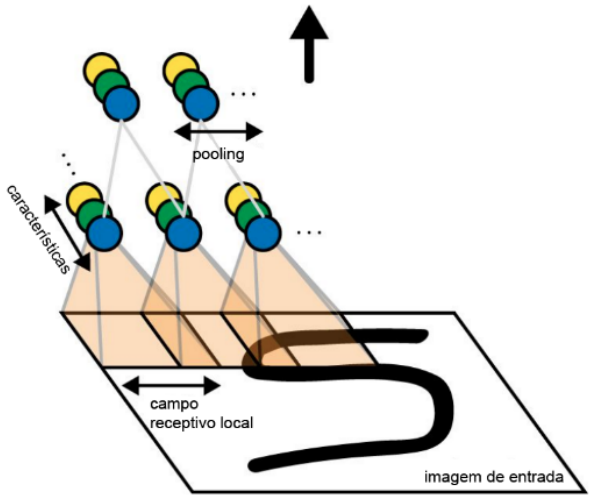

Figura 7 – Ilustração do campo receptivo local. Esse campo se desloca através da imagem que recebe as informações e processa através das camadas de convolução. Imagem adaptada de Ramsundar e Zadeh (2018)

de entrada e para cada posicionamento, a camada convolucional aplica uma multiplicação para retornar um único número correspondente ao respectivo pedaço da imagem, conforme a Figura 8. Essa multiplicação, definida na Equação 2.3, é feita pelos valores recebidos pelo campo por uma matriz normalmente quadrada, chamada de filtro K (*kernel*) convolucional (RAMSUNDAR; ZADEH, 2018), .

$$(K * I)(i, j) \;=\; \sum_m \sum_n K(m, n) I(i - m, j - n) \tag{2.3}$$

A camada convolucional pode utilizar um conjunto de n filtros convolucionais, com tamanho $p \times q$. Os melhores valores, para cada filtro, são aprendidos durante o treinamento da rede. Entretanto, a quantidade n e tamanho $p \times q$ devem ser especificados para o algoritmo. Assim, cada filtro K_l, com $l = 1, \ldots, n$, é aplicado a imagem I, produzindo um conjunto de mapas de características de tamanho n. Esse conjunto de mapas de características são empilhados, formando a saída da camada de convolução. Conforme os filtros percorrem a imagem de entrada, a quantidade de passos dados por um filtro para capturar um novo campo receptivo local é chamada de *stride* (GOODFELLOW et al., 2016). Por exemplo, na Figura 8, o campo receptivo no tempo inicial, representado em vermelho, desloca-se uma casa para a direita na próxima unidade de tempo, em azul. Logo, tem-se $stride = 1$.

Os filtros percorrem a imagem da esquerda para a direita, de cima para baixo. Logo, dependendo do tamanho da imagem, do tamanho do filtro e *stride* definidos, alguns *pixels* da imagem podem ser perdidos. Como normalmente os *kernels* são pequenos, para qualquer convolução, somente alguns *pixels* são perdidos. Porém, essa perda de informação escala conforme avança para outras camadas de convolução. Uma solução é adicionar

pixels extras, geralmente com o valor 0, para preencher as bordas da imagem de entrada, aumentando o tamanho da imagem. Essa técnica é conhecida como *padding* (Figura 8).

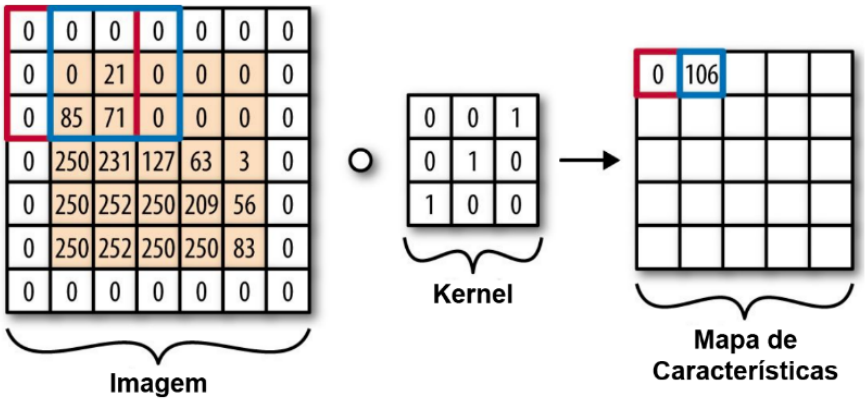

Figura 8 – O processo de convolução. Os *kernels* percorrem a imagem de entrada, executando a multiplicação para cada estado do campo receptivo local, para produzir os mapas de características. A zona laranja representa a imagem original e os 0 ao redor representam o *padding* aplicado. Imagem adaptada de Ramsundar e Zadeh (2018)

Seguindo o processamento da rede, a camada de *pooling* recebe as características extraídas pela camada convolucional e executa uma operação para reduzir o tamanho destas características e, consequentemente, o custo computacional do treinamento da RNC (RAMSUNDAR; ZADEH, 2018). Diversas operações podem ser utilizadas, mas a mais conhecidas é o *max pooling*, definido na Equação 2.4, que fornece como saída o valor máximo de uma vizinhança retangular r do mapa de entrada, conforme a Figura 9.

$$Pooling(a, b) = max\{(K * I)(a', b'),\ a' \in [a, a + r],\ b' \in [b, b + r]\} \tag{2.4}$$

Após várias execuções das camadas de convolução e de *pooling*, as características extraídas são passadas para a camada totalmente conectada. Nesta etapa temos uma rede neural totalmente conectada com uma ou mais camadas. Essa rede é chamada de totalmente conectada porque todos os neurônios de uma camada recebem os valores de todos os neurônios da camada anterior para gerar uma informação semântica (GU et al., 2018). Assim como descrito na subseção 2.1.1, a rede totalmente conectada utiliza funções de ativação entre suas camadas e pode utilizar também o Dropout e o BN.

Por fim, depois da camada totalmente conectada está a camada de saída. A camada de saída possui neurônios com funções de ativação, cujo objetivo é transformar os valores contínuos da camada totalmente conectada em valores que se adéquam ao objetivo da rede. Por exemplo, para classificação binária, geralmente é utilizada uma camada de saída com um neurônio aplicado à função Sigmoid, ilustrada na Figura 3, e para classificação com

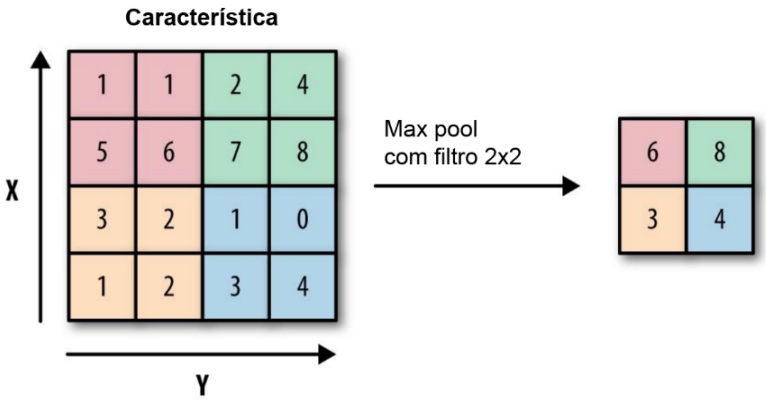

Figura 9 – O processo de *pooling*. Os mapas de características são reduzidos através de alguma função, nesse caso, o *max pooling*, que percorre o mapa e produz como saída o maior valor encontrado em cada região, representadas pelas cores vermelha, verde, amarela e azul. Imagem adaptada de Ramsundar e Zadeh (2018)

mais de duas classes é utilizada uma camada de saída com o mesmo número de neurônios que o número de classes, aplicados então à função Softmax (GOODFELLOW et al., 2016).

Além do aumento da capacidade computacional ao longo dos anos, outros fatores ajudaram a impulsionar a evolução das RNCs, como: conjuntos de dados disponibilizados publicamente com grandes quantidades de imagens rotuladas, que é necessário para o treino das RNCs, para os mais diversos problemas; e competições que foram criadas para medir a performance das redes, sendo a ImageNet Large Scale Visual Recognition Challenge (ILSVRC) uma das competições mais importantes.

A ILSVRC foi uma competição anual online, realizada entre 2010 e 2017, no campo da visão computacional, com o objetivo de promover o desenvolvimento de técnicas para detecção e localização de objetos e classificação de imagens, além de servir como referência dos algoritmos estado-da-arte (RUSSAKOVSKY et al., 2015). Na parte de classificação de imagens do campeonato, normalmente os participantes da competição recebem três conjuntos de dados, sendo eles: o conjuntos de dados de treino, que é utilizado para gerar o modelo; o conjuntos de dados de validação, que é utilizado para garantir que o modelo é capaz de generalizar bem em novos dados; e o conjuntos de dados de teste, onde somente a organização do evento possui os rótulos, sendo onde os modelos serão efetivamente avaliados após serem submetidos (RUSSAKOVSKY et al., 2015). Esses conjuntos de dados compreendem aproximadamente um milhão de imagens e mil classes de objetos diferentes (RUSSAKOVSKY et al., 2015). Além disso, esses conjuntos de dados são subconjuntos, sem intersecção, de um conjuntos de dados ainda maior, o ImageNet (DENG et al., 2009), que contém mais de 14 milhões de imagens. O ImageNet é disponibilizado publicamente, assim como os conjuntos de dados utilizados em cada ano da ILSVRC.

Algumas das arquiteturas de RNCs que obtiveram destaque no ILSVRC para

classificação de imagens, são: AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2017), campeã da edição 2012, possui uma arquitetura parecida com a LeNet, porém mais profunda, maior e com mais camadas convolucionais; ZFNet (ZEILER; FERGUS, 2014), campeã da edição 2013, aumentou a acurácia da AlexNet através do uso de mais camadas intermediárias; GoogleNet (SZEGEDY et al., 2015), campeã da edição 2014, apresentou um novo módulo que reduziu drasticamente o número de parâmetros na rede; ResNet (HE et al., 2016), campeã da edição 2015, apresentou novos tipos de conexões, de normalização e da estrutura final da rede; GDB-Net (ZENG et al., 2017), campeã da edição 2016, apresentou um framework composto de redes convolucionais; SENet (HU; SHEN; SUN, 2018), campeã da edição 2017, propôs uma nova unidade na arquitetura da rede chamada de *Squeeze-and-Excitation* com o objetivo de recalibrar as características extraídas da convolução.

Entre as arquiteturas participantes no ILSVRC, também se destaca a VGG (SIMONYAN; ZISSERMAN, 2014), que foi campeã na tarefa de localização e vice campeã na tarefa de classificação de imagens em 2014. Ela possui uma arquitetura formada por uma sequência de camadas de convolução com filtros de tamanho 3×3 seguidas de uma camada de *max pooling*, executadas de forma consecutivas (SIMONYAN; ZISSERMAN, 2014).

Segundo Khan et al. (2020), a maioria das arquiteturas atuais, inclusive a ResNet (HE et al., 2016), são construídas em cima dos princípios de topologia simples e homogênea, apresentada pela VGG. Essa topologia pode ter vários tamanhos diferentes, o que é indicado no nome da arquitetura, por exemplo: a VGG19 (SIMONYAN; ZISSERMAN, 2014) possui 19 camadas e a VGG16 (LIU; DENG, 2015) possui 16 camadas. As duas topologias estão representadas na Figura 10, onde FC representa uma rede totalmente conectada e as camada de *pooling* não são contadas no número de camadas.

Devido ao tamanho da arquitetura e o tamanho do conjunto de dados de treinamento, essas redes trabalham na otimização de milhares de parâmetros em seu treinamento, resultando em um processo extremamente custoso em termos de *hardware* necessário e de tempo para execuções. Por isso, depois de treinadas, essas redes geralmente são disponibilizadas publicamente, sendo possível utilizá-las em outras tarefas após um processo de transferência de aprendizado (YOSINSKI et al., 2014). Segundo Yosinski et al. (2014), as primeiras camadas das RNCs aprendem características gerais das imagens, enquanto as últimas camadas aprendem características mais específicas. Assim, a transferência de aprendizado permite que um modelo f já treinado para um problema X, pode ser ajustado para outro problema Y. Para isso, deve-se manter as camadas gerais de f e modificar, ou aumentar, as camadas específicas, permitindo aproveitar o conhecimento geral que f já possui e retreinar somente as novas camadas específicas para o novo problema Y (YOSINSKI et al., 2014). O processo de retreinar somente as camadas específicas é conhecido

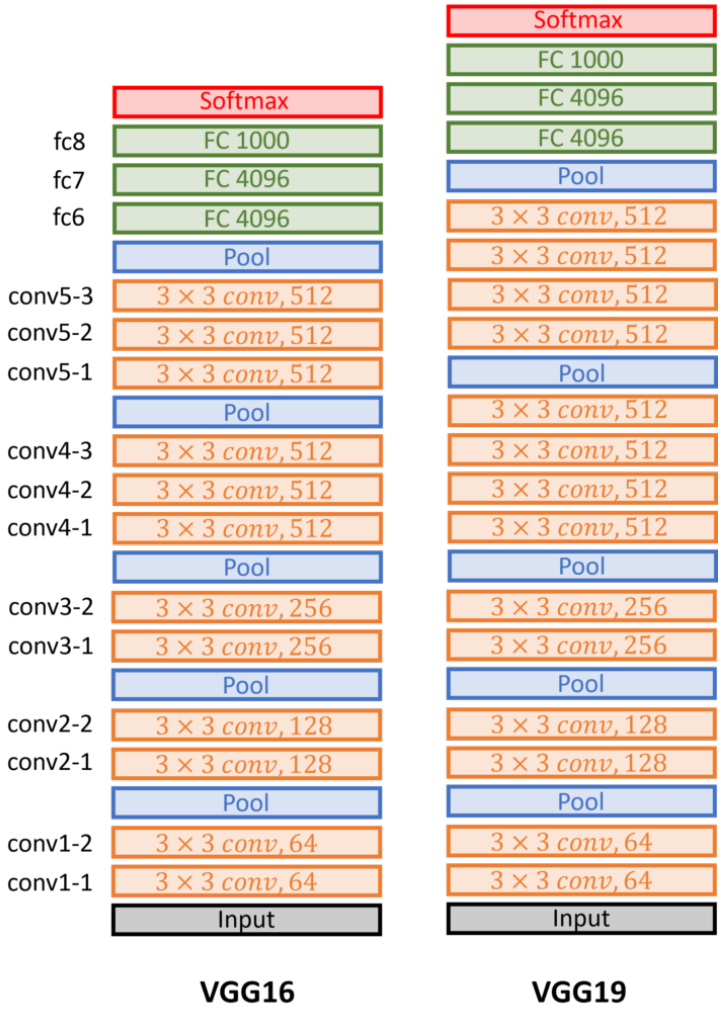

Figura 10 – Topologia da VGG16 e da VGG19. A VGG16, representada na esquerda, possui 16 camadas, sendo: 12 camadas convolucionais, 3 camadas totalmente conectadas e 1 camada de saída. A VGG19, representada na direita, possui 19 camadas, sendo: 15 camadas convolucionais, 3 camadas totalmente conectadas e 1 camada de saída. Imagem disponível em <http://datahacker.rs/deep-learning-vgg-16-vs-vgg-19>

como *finetunning* (GOODFELLOW et al., 2016).

Yosinski et al. (2014) e Razavian et al. (2014) realizaram muitos experimentos sobre transferência de aprendizado, demonstrando que: as redes possuem, de fato, a capacidade de transferir aprendizado; o tempo necessário para treinar redes com transferência de aprendizado é menor do que o tempo necessário para treinar uma rede do zero com pesos aleatórios; e a quantidade de imagens necessárias para o novo treinamento é menor. Com isso, uma empresa, pesquisador ou entusiasta pode utilizar uma RNC pré-treinada em um conjunto de dados complexo, como utilizado no ILSVRC, para transferir esse conhecimento geral para um novo modelo que atenda as necessidades do problema que desejam resolver.

Algumas arquiteturas de RNCs também apresentam resultados estado-da-arte em

problemas de aprendizado de máquina não supervisionado, como as Redes Generativas Adversárias (RGAs), que pertencem aos algoritmos generativos. O objetivo dos algoritmos generativos é, a partir de dados de treino $\sim p_{data}(x)$ oriundos de uma distribuição desconhecida, gerar novas amostras de dados $\sim p_{model}(x)$ da mesma distribuição (JABBAR; LI; OMAR, 2020).

2.1.3 Redes Generativas Adversárias

Proposto por Goodfellow et al. (2014), a Rede Generativa Adversária é um tipo de algoritmo generativo que usa duas redes neurais, competindo entre si, como adversárias, conforme mostra a Figura 11. A primeira rede, conhecida como Gerador (G), recebe valores aleatórios retirados de uma distribuição normal como entrada e gera imagens como saída, com o objetivo de enganar o Discriminador. A segunda rede, conhecida como Discriminador (D), possui a tarefa de classificar corretamente quais imagens são verdadeiras $\sim pdata(x)$ e quais foram geradas por G (GOODFELLOW et al., 2014).

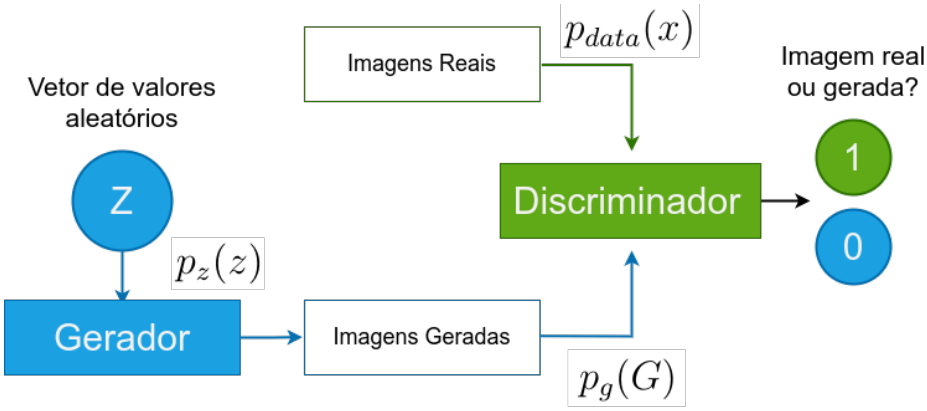

Figura 11 – Visão geral da Rede Generativa Adversária. O Gerador recebe um vetor de valores aleatórios z, retirados de uma distribuição uniforme e produz imagens sintéticas. O Discriminador recebe as imagens reais e as imagens sintéticas para decidir quais são verdadeiras ou não. Essa interação entre o Gerador e o Discriminador tem o objetivo de aproximar a distribuição p_g da distribuição original p_{data}. Imagem adaptada de Jabbar, Li e Omar (2020)

Dessa forma, para que G aprenda a distribuição $p_{model} = p_g$ sobre os dados x, define-se a variável aleatória de entrada como $p_z(z)$ e, em seguida, um mapeamento é representado para o espaço de dados como $G(z, \theta_g)$, onde G é uma função diferenciável representada por uma rede com parâmetros θ_g. O Discriminador produz um único valor escalar, onde $D(x)$ representa a probabilidade de x ter sido retirado de p_{data} ao invés de p_g. Então, treina-se D para maximizar a probabilidade de atribuir o rótulo correto para exemplos retirados do conjunto de dados de treino e de $G(z, \theta_g)$. Enquanto, de forma simultânea, G é treinada para minimizar o $\log(1 - D(G(z)))$, de acordo com a Equação 2.5

(GOODFELLOW et al., 2014).

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.5}$$

A Figura 12 apresenta o processo de treino das RGAs através da atualização simultânea da distribuição de D (linha tracejada azul) para discriminar entre exemplos retirados de $p_{data}(x)$ (linha pontilhada preta) dos exemplos gerados por G (linha sólida verde). A linha horizontal inferior representa o domínio de onde z é amostrado, nesse caso, uniformemente. A linha horizontal superior é a parte do domínio de x. As setas demonstram como o mapeamento $x = G(z)$ resulta na distribuição p_g (GOODFELLOW et al., 2014).

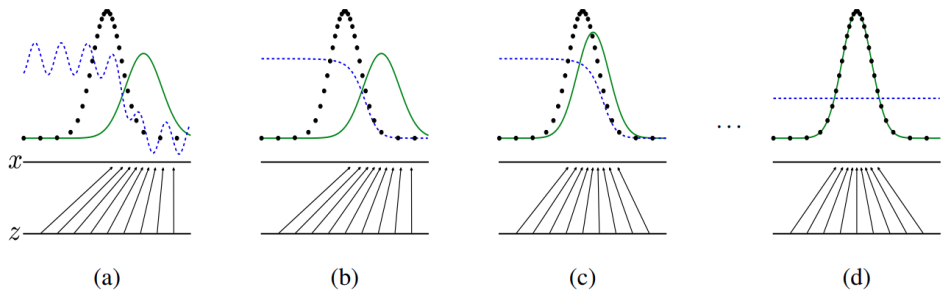

Figura 12 – O processo de treino da Rede Generativa Adversária. (a) Considerando uma RGA próxima da convergência: p_g é similar a p_{data} e D é um classificador parcialmente preciso. (b) Dentro do *loop* interno do algoritmo da RGA, D é treinado para discriminar os exemplos verdadeiros, convergindo para $D'(x) = \frac{p_{data}(x)}{p_{data}(x)+p_g x}$. (c) Depois de atualizar G, o gradiente de D guia $G(z)$ para regiões onde a probabilidade das imagens serem consideradas verdadeiras é maior. (d) Depois de várias etapas de treinamento, G e D, caso possuam capacidade suficiente, vão convergir para um ponto onde $p_g = p_{data}$ e o discriminador será incapaz de diferenciar entre as duas distribuições, resultando em $D(x) = \frac{1}{2}$ (GOODFELLOW et al., 2014).

Antes das RGAs, outros modelos generativos foram propostos, como *Deep Belief Networks* (HINTON; OSINDERO; TEH, 2006), *Stacked Convolutional Auto-Encoders* (MASCI et al., 2011) e *Deep Generative Stochastic Network* (BENGIO et al., 2014). Entretanto, o trabalho de Goodfellow et al. (2014) apresentou resultados melhores e, desde então, as RGAs são o estado da arte em modelos generativos.

A Figura 13 mostra a topologia básica das RGAs, onde o Discriminador e o Gerador são RNCs convencionais. A única diferença é que G recebe uma entrada de dimensão pequena e gera como saída uma imagem de dimensão maior. Para isso, G utiliza camadas convolucionais transpostas (ZEILER et al., 2010) no lugar das camadas convolucionais tradicionais, descritas na subseção 2.1.2. Essa camada é a transformação inversa a camada convolucional tradicional (DUMOULIN; VISIN, 2016).

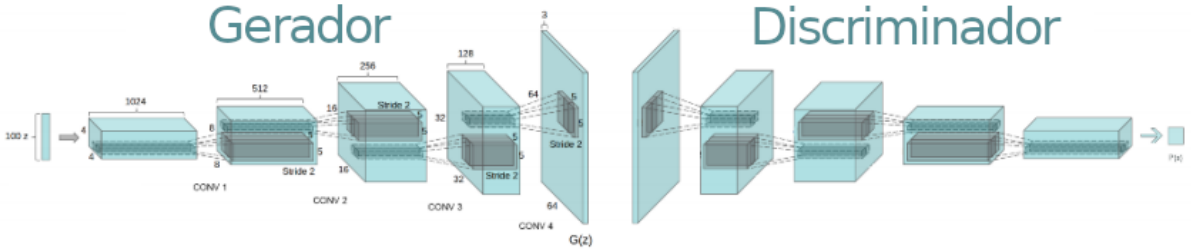

Figura 13 – Topologia da Rede Generativa Adversária. O Gerador é composto por camadas de convolução transposta e o Discriminador possui a arquitetura de uma RNC convencional. Imagem adaptada de BASART (2017)

Um problema comum durante o treinamento das RGAs é o *mode collapse*. Nele, G começa a ignorar o espaço de entrada e produzir somente algumas mesmas imagens de saída, conforme a distribuição azul na Figura 14. O que acontece é que $G(z)$ gera como saída a mesma imagem I_1 para diferentes vetores latentes oriundos de uma distribuição normal Z, ou seja, $G(\{z_i\}_{i=1}^{k}) = I_1$. Quanto mais próximo forem os vetores latentes de entrada, por exemplo z_1 e z_2, mais as imagens de saída $I_1 = G(z_1)$ e $I_2 = G(z_2)$ tendem a colapsar na mesma moda (MAO et al., 2019).

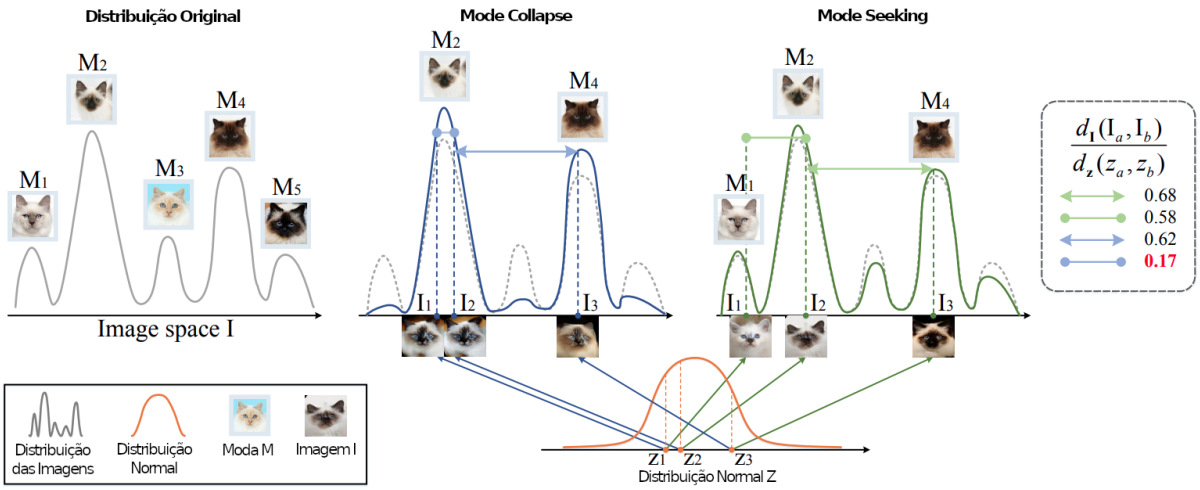

Figura 14 – Ilustração do problema de *mode collapse*. A linha cinza sólida e pontilhada representa a distribuição original p_{data} e desconhecida das imagens reais. A linha azul representa a distribuição do Gerador p_g quando o *mode collapse* ocorre, onde vetores próximos são mapeados para a mesma imagem. A linha verde representa a distribuição do Gerador p_g quando o regularizador MS é aplicado, forçando que as imagens produzidas a partir de vetores próximos sejam diferentes. Imagem adaptada de Mao et al. (2019)

Com o objetivo de diminuir o *mode collapse*, Mao et al. (2019) apresentou uma regularização, denominada Mode Seeking (MS) e definida por $\mathcal{L}_{ms}$ na Equação 2.6, que calcula a distância euclidiana $d(.)$ das imagens produzidas por G, dividido pela distância euclidiana dos respectivos vetores de entrada. Para controlar a importância da regularização

introduzida por MS, $\mathcal{L}_{ms}$ é multiplicada por um peso λ_{ms}. Antes de $\mathcal{L}_{ms}$ ser adicionada a função de erro $\mathcal{L}_{original}$, produzindo a nova função de perda $\mathcal{L}_{nova}$ (Equação 2.7), $\mathcal{L}_{ms}$ é elevada a -1 para inverter o relacionamento das distâncias, uma vez que o objetivo do *backpropagation* é minimizar o erro e quanto maior $\mathcal{L}_{ms}$, melhor G é em produzir imagens diferentes. O trabalho de Mao et al. (2019) se destaca entre os demais pela simplicidade e pelos resultados qualitativos e quantitativos apresentados, que apresentam a efetividade em melhorar a diversidade das imagens de saída sem a perda de qualidade.

$$\mathcal{L}_{ms} = \max_{G}\left(\frac{d_I(G(z_1), G(z_2))}{d_z(z_1, z_2)}\right) \tag{2.6}$$

$$\mathcal{L}_{nova} = \mathcal{L}_{original} + \lambda_{ms}\mathcal{L}_{ms}^{-1} \tag{2.7}$$

Um problema ainda em aberto no desenvolvimento de RGAs é a falta de uma função ou métrica capaz de avaliar a qualidade das imagens geradas e permitir a comparação entre diferentes RGAs. Atualmente, a função de perda, apresentada na Equação 2.5 representa um reflexo do quão bem o Gerador é capaz de enganar o Discriminador ou o Discriminador é preciso ao diferenciar imagens verdadeiras ou geradas. Entretanto, otimizar essa função não garante que as imagens geradas sejam realistas (BASART, 2017). A métrica mais utilizada na literatura para otimizar o Gerador, deixando as imagens mais realistas é o Inception Score (IS) (SALIMANS et al., 2016), definida na Equação 2.8. O IS utiliza um modelo de classificação de imagens M pré treinado no ImageNet (DENG et al., 2009), conhecido como *Inception Network* (SZEGEDY et al., 2016), em sua métrica, onde $pM(y|x)$ representa a distribuição de x predito por M e $pM(y) = \int_x pM(y|x)d\mathbb{P}_g$. Um IS mais alto significa que a *Inception Network* é mais confiante de que as imagens pertencem a alguma categoria da ImageNet e possuem qualidade e diversidade (XU et al., 2018). Salimans et al. (2016) demonstrou que o IS possui uma correlação razoável com o julgamento humano.

$$\mathrm{IS}(\mathbb{P}_g) = e^{\mathbb{E}_{x\sim\mathbb{P}_g}[KL(pM(y|x)||pM(y))]} \tag{2.8}$$

Outra métrica utilizada é a Fréchet Inception Distance (FID), introduzido por Heusel et al. (2017), é utilizada para calcular a distância entre a qualidade das imagens geradas pelo Gerador e as imagens originais, onde quanto menor a FID, melhor o Gerador. Essa distância de qualidade e diversidade é calculada utilizando a média e a covariância dos vetores de saída da última camada convolucional da *Inception Network* para as imagens geradas e originais (XU et al., 2018). A Figura 15 demonstra como a FID se comporta em diversos casos de pertubação. Essa também foi a métrica utilizada por Mao et al. (2019) para demonstrar a eficácia do MS. O IS e a FID são as principais métricas utilizadas na literatura dentre outras, como o *LPIPS* (ZHANG et al., 2018), o *Mode Score*, o *Kernel MMD*, e o *Wasserstein Distance* (XU et al., 2018).

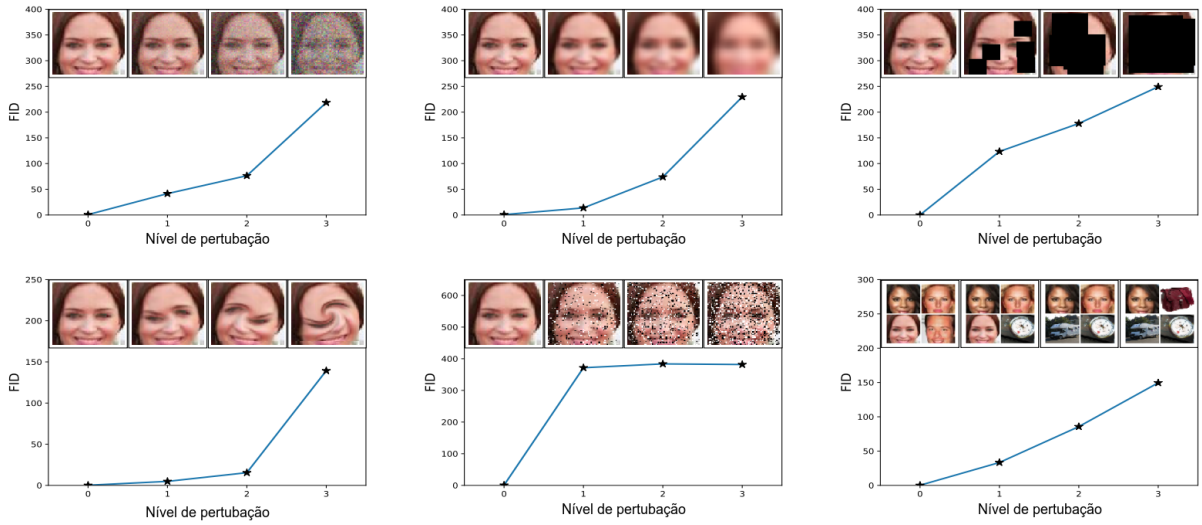

Figura 15 – Correlação do FID com diferentes tipos e níveis de pertubação. As imagens da parte superior, da esquerda para direita, sofreram os seguintes tipos de pertubação: ruído gaussiano, borrão gaussiano e retângulos negros implantados. As imagens da parte inferior, da esquerda para direita, sofreram os seguintes tipos de pertubação: rotacionamento interno, ruído e adição de imagens da ImageNet (DENG et al., 2009). O FID é capaz de capturar muito bem todos os tipos de pertubação e os diferentes níveis aplicados. Imagem adaptada de Heusel et al. (2017)

A primeira RGA apresentada no trabalho do Goodfellow et al. (2014) é chamada de Vanilla GAN (BASART, 2017), onde GAN (*Generative Adversarial Network*) é a sigla em inglês para RGA. Após a Vanilla GAN, outras arquiteturas foram propostas e as principais estão representadas na Figura 16. As arquiteturas podem ser agrupadas em duas categorias: (a) onde o Discriminador possui acesso ao rótulo ou à variável latente antes de realizar a predição; e (b) onde o Discriminador é quem prevê o rótulo (BASART, 2017).

A Conditional GAN, apresentada por Mirza e Osindero (2014), é uma arquitetura onde a saída do gerador está condicionada ao rótulo da classe desejada. O Discriminador recebe como entrada a imagem gerada pelo Gerador e o rótulo para decidir se a imagem gerada faz parte da distribuição original e se a imagem gerada pertence a classe desejada. A Bidirectional GAN, apresentada por Donahue, Krähenbühl e Darrell (2016), possui uma rede separada, cujo objetivo é gerar a variável latente z para cada imagem de entrada. O Discriminador recebe tanto z quando a imagem gerada. Já a Semi-Supervised GAN (ODENA, 2016), a InfoGAN (CHEN et al., 2016) e a Auxiliary Classifier GAN (ODENA; OLAH; SHLENS, 2017) apresentam diferenças sutis, mas todas possuem um Discriminador que prevê qual a classe da imagem gerada, além do tradicional verdadeiro ou falso.

O trabalho de Yuan et al. (2020) utilizou uma Auxiliary Classifier GAN, com IS e MS, para produzir imagens sintéticas pertencentes a uma distribuição desconhecida,

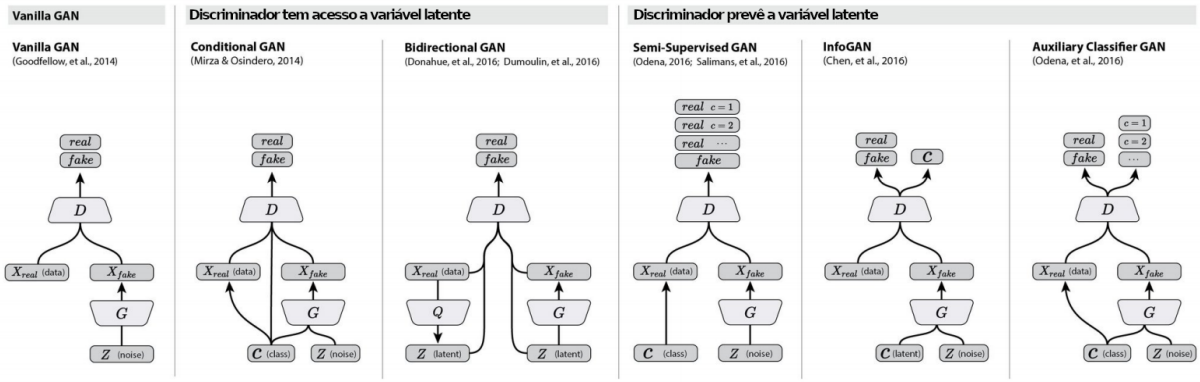

Figura 16 – Arquiteturas propostas para RGAs. A Conditional GAN (MIRZA; OSINDERO, 2014) possui a saída do gerador condicionada ao rótulo da classe desejada. O Discriminador deve decidir se a imagem gerada faz parte da distribuição original e se pertence a classe informada. A Bidirectional GAN (DONAHUE; KRÄHENBÜHL; DARRELL, 2016), possui uma rede separada, cujo objetivo é gerar a variável latente z para cada imagem de entrada. O Discriminador recebe tanto z quando a imagem gerada. A Semi-Supervised GAN (ODENA, 2016), a InfoGAN (CHEN et al., 2016) e a Auxiliary Classifier GAN (ODENA; OLAH; SHLENS, 2017) apresentam diferenças sutis e possuem um Discriminador que prevê qual a classe da imagem gerada, além do tradicional verdadeiro ou falso. Imagem adaptada de BASART (2017)

utilizada originalmente para treinar uma RNC. Essas imagens foram utilizadas extrair informações do modelo original com o objetivo de recriá-lo. Essa ação, conhecida como extração de modelos, faz parte dos ataques de privacidade que modelos de aprendizado de máquina em geral podem sofrer.

2.2 Ataques de Privacidade

Conforme os modelos de aprendizado de máquina se tornam mais amplamente utilizados, a necessidade de estudar suas implicações e vulnerabilidades na sociedade se tornam mais urgentes. Assuntos como ética (HIBBARD, 2014), viéses (CATON; HAAS, 2020), explicabilidade e interpretabilidade (ISLAM et al., 2021), segurança (WANG et al., 2019) e privacidade de modelos, tem recebido uma atenção crescente da comunidade (RIGAKI; GARCIA, 2020).

Em ataques relacionados a privacidade, o objetivo do atacante é ganhar informações que a princípio não deveriam ser compartilhadas, como: informações a respeito do conjunto de dados de treino, informações a respeito do modelo em si ou até mesmo extrair informações a respeito das propriedades do conjunto de dados de treino, como vieses que o modelo aprendeu intencionalmente (RIGAKI; GARCIA, 2020). A Figura 17 ilustra uma ideia geral do ambiente, com os diferentes atores e as informações que podem ser alvo de um ataque de privacidade.

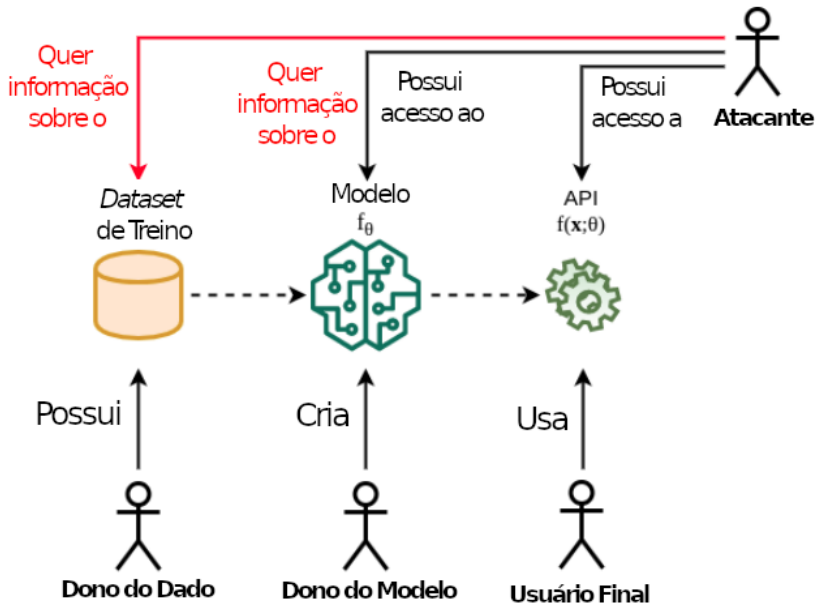

Figura 17 – Visão geral do ambiente em um ataque de privacidade. As figuras de humanos representam os diferentes atores e os outros símbolos representam as informações que precisamos proteger. Linhas pontilhadas representam o fluxo da informação e linhas sólidas representam possíveis ações dos atores, onde as linhas vermelhas são as ações que o atacante deseja realizar. Imagem adaptada de Rigaki e Garcia (2020)

Quando o atacante não possui nenhuma informação sobre o modelo alvo, somente o vetor de predição, o ataque é considerado como caixa preta. Entretanto, se o atacante conhece qual foi a arquitetura utilizada ou quais são os parâmetros ou hiperparâmetros do modelo alvo, o ataque é conhecido como caixa branca (RIGAKI; GARCIA, 2020). Por esse motivo que a Figura 17 possui uma seta vermelha e preta de ação do atacante em relação ao modelo. Além dessas classificações, os ataques de privacidade podem ser categorizados em quatro tipos, sendo eles: inferência de pertencimento; reconstrução; inferência de propriedade; e extração de modelos, que é o tipo de ataque realizado neste trabalho (RIGAKI; GARCIA, 2020).

A inferência de pertencimento é um tipo de ataque de privacidade cujo o objetivo é determinar se uma amostra foi utilizada no conjunto de dados de treino do modelo alvo. Esta é a categoria de ataques mais conhecida e foi introduzida por Shokri et al. (2017). Esse tipo de ataque também pode ser utilizado para auditar modelos (CHEN et al., 2020; HAYES et al., 2017; HILPRECHT; HÄRTERICH; BERNAU, 2019) caixa preta para checar se uma amostra foi utilizada sem o consentimento do dono, por exemplo (RIGAKI; GARCIA, 2020). Já nos ataques de reconstrução, o objetivo é recriar, de forma parcial ou completa, um ou mais exemplos do conjunto de dados de treino e/ou seus respectivos rótulos. E nos ataques de inferência de propriedade, busca-se extrair propriedades do conjunto de dados de treino que não estavam explicitamente definidas como uma *feature* ou

não estavam correlacionados com a tarefa de aprendizado desejada. Um exemplo de ataque de inferência de propriedade é a extração da razão entre mulheres e homens presentes em um conjunto de dados (RIGAKI; GARCIA, 2020).

Diferente dos demais, a extração de modelos é um tipo de ataque caixa preta, onde o atacante tenta extrair informações do modelo alvo f com o objetivo potencial de construir um modelo substituto $\hat{f}$ que se comporta de forma semelhante ($\hat{f} \approx f$)(RIGAKI; GARCIA, 2020). Para o atacante, existem dois objetivos ao criar $\hat{f}$. O primeiro objetivo, conhecido como extração de acurácia, é criar um modelo substituto $\hat{f}$ capaz de performar com a mesma acurácia que o modelo original f em um conjunto de dados que faz parte da mesma distribuição do conjunto de dados de treino de f, ou seja, um conjunto de dados que está relacionado a tarefa de aprendizado original. O segundo objetivo, conhecido como extração de fidelidade, é semelhante ao primeiro, porém em um conjunto de dados não relacionado a tarefa de aprendizado original (JAGIELSKI et al., 2020).

Para realizar a extração, o atacante executa uma série de requisições ao modelo alvo, salvando as predições de saída. Esses pares de entrada e saída podem ser vistos como um sistema de equações, onde as variáveis desconhecidas são os parâmetros e hiperparâmetros do modelo alvo (RIGAKI; GARCIA, 2020). Entretanto, em casos complexos, onde o modelo alvo é uma RNC, o sistemas de equações não é linear e para isso é necessário o uso de técnicas de otimização, como o gradiente descendente estocástico, para aproximar os parâmetros do modelo (TRAMÈR et al., 2016). Oh, Schiele e Fritz (2019) verificaram que é a probabilidade de uma extração ser bem sucedida é maior contra os modelos alvos que possuem uma acurácia de 98% ou mais no conjunto de dados de teste. Além disso, quanto pior a capacidade de generalização do modelo alvo ou quanto maior o número de classes diferentes do conjunto de dados de treino, mais difícil de efetuar um ataque de extração (LIU et al., 2021). Alguns trabalhos, mais teóricos, provaram a possibilidade de executar a extração direta de modelos além dos lineares. Milli et al. (2019) mostrou que a completa extração de redes neurais totalmente conectadas com duas camadas, utilizando a função de ativação ReLU, é teoricamente possível. Entretanto, os dois trabalhos supõem que o atacante tem acesso aos gradiente da função de perda para cada requisição. Outros trabalhos mais recentes, conseguiram executar a extração de modelos utilizando somente a classe de saída do modelo alvo (CORREIA-SILVA et al., 2021; JAGIELSKI et al., 2020).

Mesmo sendo de interesse da academia proteger tais modelos, as causas que possibilitam a extração ainda não são totalmente claras. Muitos mecanismos de defesa propostos contra ataques de extração possuem o objetivo de identificar se uma requisição realizada por um usuário é normal ou se faz parte de uma série de requisições feitas durante um ataque (RIGAKI; GARCIA, 2020). PRADA (JUUTI et al., 2019), um dos primeiros métodos de defesa, é capaz de detectar ataques de extração baseado na hipótese de que, como o atacante deseja explorar os limites de decisão do modelo alvo, as requisições

feitas em um ataque possuem uma distribuição diferente das requisições normais. A detecção proposta funciona, porém, os autores notaram que o atacante pode se adaptar a estratégia, evitando a detecção (RIGAKI; GARCIA, 2020). Krishna et al. (2019) também apresentaram um método de defesa contra ataques de extração baseado na inferência de pertencimento. O trabalho se baseia na premissa de que, através da inferência de pertencimento, o modelo é capaz de distinguir entre requisições legítimas e requisições feitas pelo atacante, onde o único propósito é a extração de informações. Os autores notaram que esse tipo de defesa possui algumas limitações, como potencialmente marcar uma requisição legítima como maliciosa (falso positivo) e atrapalhar o usuário legítimo. Esse tipo de defesa também pode ser evitado se o atacante adaptar as requisições maliciosas.

Além do objetivo de produzir um modelo substituto diretamente, um ataque de extração também possibilita a extração de outras informações do modelo alvo, como arquitetura, algoritmos de otimização e hiperparâmetros, através de modelos sombra (OH; SCHIELE; FRITZ, 2019). Alguns conjuntos de dados de domínio público são utilizados na realização de trabalhos de ataque de privacidade. Esses conjuntos servem como base para comparações entre os métodos propostos, enquanto torna mais democrático o acesso a grande volume de dados.

2.3 Conjuntos de Dados de Domínio Público

A aquisição de imagens, principalmente rotuladas, é uma tarefa custosa, porém necessária para o treinamento de modelos de aprendizado de máquina. Por isso, diversos conjuntos de dados são fornecidos na literatura, permitindo uma forma mais simples para experimentação de novas ideias. Existem diversos tipos de conjuntos de dados para processamento de imagens, fornecendo tarefas de classificação, segmentação e detecção, por exemplo. Dentre os conjuntos de dados de classificação de imagens, escopo desse trabalho, destacam-se: CIFAR-10 (KRIZHEVSKY; HINTON et al., 2009), STL-10 (COATES; NG; LEE, 2011), MNIST (LECUN et al., 1998), EMNIST (COHEN et al., 2017) e SVHN (NETZER et al., 2011).

O conjunto de dados CIFAR-10 (KRIZHEVSKY; HINTON et al., 2009) possui 60 mil imagens coloridas de tamanho 32×32 *pixels*, igualmente distribuídas em 10 classes, ou seja, 6 mil imagens para cada classe, sendo elas: avião, carro, pássaro, gato, veado, cachorro, sapo, cavalo, navio e caminhão. A Figura 18 demonstra algumas imagens de cada classe, contidas no CIFAR-10 (KRIZHEVSKY; HINTON et al., 2009). Além da versão com 10 classes, existe a versão com 100 classes, denominada CIFAR-100 (KRIZHEVSKY; HINTON et al., 2009), mas devido ao alto número de classes com poucas imagens contidas em cada um, a versão com 10 classes é mais utilizada na literatura. O STL-10 (COATES; NG; LEE, 2011) possui imagens 100 mil imagens não rotuladas e 13 mil imagens rotuladas,

sendo $1,3$ mil imagens para cada uma das 10 classes. As imagens são coloridas e possuem tamanho 96×96 *pixels*. Além disso, as imagens do STL-10 (COATES; NG; LEE, 2011) possuem as mesmas classes do CIFAR-10 (KRIZHEVSKY; HINTON et al., 2009), menos a classe sapo, que foram substituídas por imagens da classe macaco.

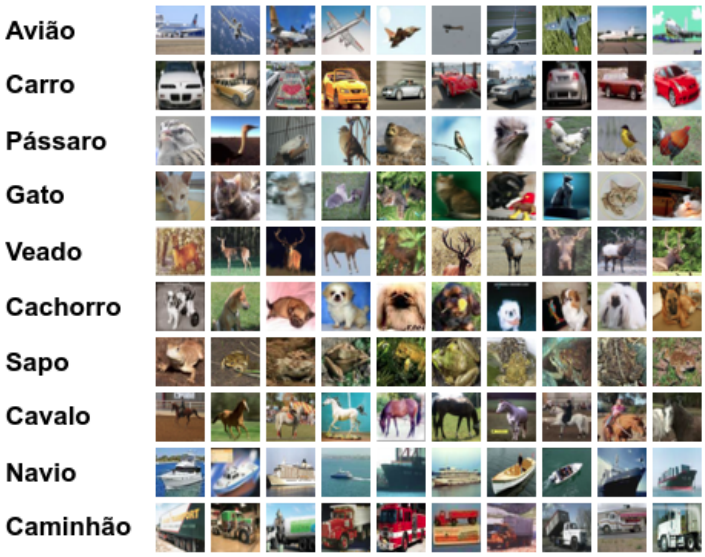

Figura 18 – Exemplo de imagens do CIFAR-10 para cada uma das 10 classes. Imagem adaptada de <https://www.cs.toronto.edu/~kriz/cifar.html>

O MNIST é um conjunto de imagens, desenvolvido por LeCun et al. (1998), que consiste na classificação de dígitos manuscritos de zero à nove. Este conjunto de dados possui 70 mil imagens de tamanho 28×28 *pixels*, com fundo preto e dígitos brancos escritos a mão. As imagens possuem os dígitos normalizados e centralizados (Figura 19). Algumas extensões do MNIST podem ser encontradas na literatura, como por exemplo, o EMNIST (COHEN et al., 2017), que possui 280 mil imagens para classificação de dígitos e 145 mil imagens para classificação de letras. O formato e estrutura das imagens é o mesmo do MNIST (LECUN et al., 1998).

Figura 19 – Exemplo de imagens do MNIST para cada uma das 10 classes (PAPERNOT et al., 2017)

Outro conjunto de dados relacionado a essas tarefas é o SVHN (NETZER et al., 2011), que consiste em imagens de números de casas utilizadas para classificação de números de zero a nove (Figura 20). Esse conjunto foi obtido a partir do *Google Street View*, possuindo 99.289 imagens no conjunto de dados principal e 531.131 imagens em um conjunto de dados extra. Cada imagem possui tamanho 32×32 pixels e representa um número da casa vista da rua, com 10 classes, de 0 a 9.

Figura 20 – Exemplo de imagens do SVHN para cada uma das 10 classes. Imagem adaptada de <http://ufldl.stanford.edu/housenumbers/>

Esses conjuntos de dados, por possuírem imagens pequenas já tratadas adequadamente para utilização, são conjuntos bons para sistemas que requerem alto custo computacional, isto é, que necessitam de muitos recursos de memória e processamento, além de poder demorar horas ou dias de treinamento. Por fim, outro conjunto de dados de destaque (já citado na subseção 2.1.2) é o ImageNet (DENG et al., 2009), que possui mais de um milhão de imagens de tamanhos variados e distribuídas em mais de mil classes diferentes. O ImageNet é utilizado em competições online, como a ILSVRC (RUSSAKOVSKY et al., 2015), e as principais arquiteturas para RNCs da literatura possuem modelos pré-treinados nele, disponíveis na Internet.

3 Trabalhos Relacionados

Alinhado ao crescimento da utilização de modelos de aprendizado de máquina e a necessidade de segurança, alguns trabalhos foram publicados explorando ataques de privacidade contra modelos alvo construídos pelos mais diversos tipos de algoritmos de aprendizado de máquina. Este capítulo apresenta uma visão geral desses trabalhos, especificamente dos trabalhos que tratam de ataques de extração de modelos, por estarem relacionados com o tema deste trabalho.

Papernot et al. (2017) utilizaram uma RNP caixa preta para classificar uma versão aumentada do conjunto de dados MNIST (LECUN et al., 1998), gerando um novo conjunto de dados contendo as imagens aumentadas e o rótulo da classificação gerada pela RNP alvo. Esse novo conjunto de dados foi utilizado para treinar uma RNP substituta, com o intuito principal de aproximar as fronteiras de decisão da RNP original, para produzir imagens adversariais contra o modelo alvo ao invés de tentar copiá-lo. Por fim, o modelo alvo era utilizado para rotular as novas imagens e todos os modelos são retreinados até atingir uma capacidade suficiente de geração de imagens adversárias. Em um novo trabalho, Papernot, McDaniel e Goodfellow (2016) apresentaram novos experimentos, também com o MNIST, onde realizaram a transferência de conhecimento de modelos classificadores de diferentes algoritmos, como RNP, regressão logística, Support Vector Machines (SVM), árvore de decisão e vizinhos próximos, para uma RNP, novamente para imitar as fronteiras de decisão do classificador original. Os modelos substitutos foram capazes de aproximar de 77% a 83% da acurácia no conjunto de dados de teste, menos para a árvore de decisão, cujo o modelo substituto foi capaz de aproximar 48%. Os dois trabalhos precisam de imagens do domínio do problema e se limitaram a aplicar as abordagens somente com o MNIST (LECUN et al., 1998).

O trabalho de Shi, Sagduyu e Grushin (2017) construiu RNPs classificadoras como modelos substitutos, utilizando um conjunto de dados com rótulos dos modelos alvos, que foram capazes de copiar 97.9% e 97.44% da performance dos modelos alvos. Os modelos alvos foram treinados pelos autores com um conjunto de dados binário para classificar texto, um modelo com o algoritmo de *Naive Bayes* e outro com SVM. Entretanto, o trabalho não explorou a extração de RNPs e requer, para sua execução, imagens que fazem parte do domínio do problema.

Em Tramèr et al. (2016), os autores abordaram a extração de modelos via API para treinar modelos substitutos com o objetivo de atingir uma performance próxima ou equivalente ao modelo alvo. O método parte da premissa de que o atacante possui acesso às probabilidades de predição do modelo para cada entrada, o que pode dificultar

o uso do método proposto em modelos servidos via API que retornam apenas o rótulo puro. O método se mostrou capaz de copiar árvores de decisão, regressões logística, SVM e Perceptron multicamadas, através de consultas as APIs dos modelos alvos hospedados em nuvem no *BigML* e no *Amazon Machine Learning*. O método proposto requer imagens do domínio de problema e possui um custo elevado, o que pode inviabilizar o uso contra RNPs.

Correia-Silva et al. (2018) investigaram ataques de extração entre modelos de aprendizagem profunda usando imagens aleatórias naturais. Em seu trabalho, o modelo alvo e o modelo substituto são RNCs, sendo aplicados em três problemas diferentes: (*i*) reconhecimento de expressão facial, (*ii*) classificação de objetos gerais e (*iii*) classificação de faixa de pedestres por imagens de satélite. O ataque Copycat, como o método é chamado, é o que mais se adéqua as restrições que um atacante encontra em um cenário real, onde o modelo alvo é uma caixa preta, disponível através de uma API, que retorna somente os rótulos específicos para cada classe do problema e nenhum conhecimento prévio do conjuntos de dados, arquitetura ou hiper-parâmetros do modelo é conhecido pelo atacante.

Nos experimentos realizados pelos autores, ocorreram a extração dos rótulos do modelo alvo para imagens do domínio de problema e para imagens naturais aleatórias, sendo o primeiro trabalho a demonstrar que a extração pode ser realizada apenas com imagens naturais aleatórias. Embora essas imagens estejam disponíveis em abundância, não há garantia que o modelo alvo produza rótulos para todas as categorias desejadas de extração do modelo alvo, sendo necessário que o atacante utilize uma diversidade de imagens para extrair os rótulos e, assim, explorar com sucesso o espaço de classificação do modelo alvo. A Figura 21 demonstra a topologia do ataque. Além disso, eles demonstraram que o método é factível no mundo real, realizando um ataque de extração com sucesso contra a *Microsoft Azure Emotion API*.

Em um novo trabalho, Correia-Silva et al. (2021) realizaram experimentos novos e mais completos com a Copycat, como a análise do espaço de entrada das imagens dos modelos alvos, o impacto das arquiteturas dos modelos alvo e cópia na acurácia de transferência, a análise qualitativa e novos problemas. Correia-Silva et al. (2021) ainda verificou o impacto de ataques realizados com 100 mil, 500 mil, 1 milhão, $1,5$ milhão e 3 milhões de requisições via API, demonstrando que quanto maior o número de requisições utilizadas na extração de conhecimento, melhor a qualidade do modelo substituto $\hat{f}$. Esses experimentos contribuiriam para um melhor entendimento das capacidades e limitações de se utilizar imagens naturais aleatórias e somente rótulos puros para extrair um modelo.

Em um trabalho similar, Orekondy, Schiele e Fritz (2019) apresentaram as redes Knockoff, onde um atacante pode roubar funcionalidades de modelos alvo caixa-preta. Assim como o trabalho inicialmente apresentado pela Copycat (CORREIA-SILVA et al., 2018), os autores utilizaram um conjunto de imagens naturais aleatórias para efetuar o

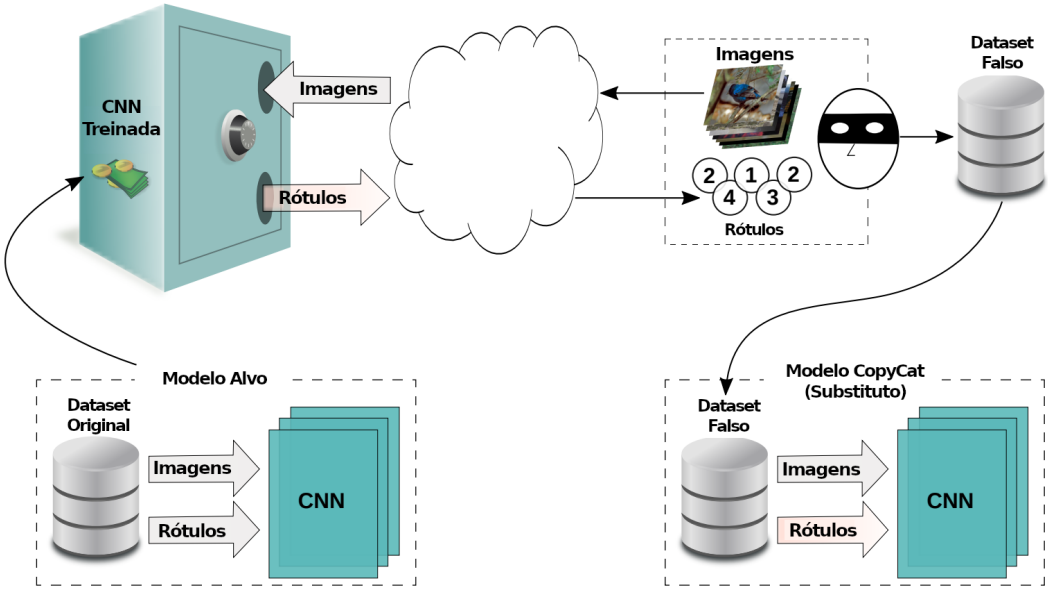

Figura 21 – Visão geral do ataque Copycat. Na esquerda, o modelo alvo, treinado com o conjunto de dados original e confidencial, é servido publicamente via API, recebendo imagens como entrada e providenciando os rótulos como saída. Na direita, é apresentado o processo do ataque de extração para criar um conjunto de dados falso, onde as imagens naturais são enviadas para a API do modelo alvo e são anotadas com os rótulos recebidos. Por fim, a rede Copycat é treinada com o conjunto de dados falso (CORREIA-SILVA et al., 2018). Imagem adaptada de Correia-Silva et al. (2018)

ataque. Porém, foram utilizadas as probabilidades das imagens pertencerem a cada uma das classes, ao invés do rótulo único, para gerar o conjunto de dados falso e treinar o modelo substituto. Fornecer as probabilidades como respostas das consultas à API trazem mais informações para o atacante, mas podem ser facilmente evitada pelos proprietários do modelo alvo, dificultando qualquer ataque desse tipo. Os experimentos foram conduzidos para quatro problemas distintos com modelos substitutos pré-treinados com arquiteturas diferentes. Para avaliar a influência da arquitetura, os modelos alvo foram gerados com duas arquiteturas diferentes, VGG-16 e ResNet-34 para cada problema. Por fim, os autores concluíram que é melhor utilizar arquiteturas mais complexas para o modelo cópia do que a arquitetura do modelo alvo, alcançando mais de 77% de cópia nos experimentos.

Outro método de extração foi apresentado em Mosafi, David e Netanyahu (2019a), onde os autores utilizaram as probabilidades de modelos alvo treinados nos conjunto de dados MNIST (LECUN et al., 1998) e CIFAR-10 (KRIZHEVSKY; HINTON et al., 2009) para rotular dados não rotulados. Assim como em Orekondy, Schiele e Fritz (2019), o uso das probabilidades de cada rótulo, apesar de fornecer mais informações e permitir uma melhor execução da extração, pode ser facilmente evitada pelo modelo alvo, uma vez que ele retorne somente o rótulo único. Em um novo trabalho, Mosafi, David e Netanyahu (2019b) prôpos um novo método para gerar imagens não rotuladas, através da composição

de duas imagens diferentes, ou seja, para duas imagens I_1 e I_2, retiradas de conjuntos de dados públicos, como o ImageNet (DENG et al., 2009), a composição dessas imagens cria uma nova imagem I_{nova}, onde $I_{nova} = \alpha * I_1 + (1 - \alpha) * I_2$, sendo $\alpha \in [0,0; \ldots; 1,0]$ a proporção da imagem I_1 e $1 - \alpha$ a proporção da imagem I_2. Essa técnica de composição é utilizada para gerar milhões de imagens diferentes não rotuladas, que por sua vez, são utilizadas para extrair rótulos dos modelos alvos e compor o conjunto de dados utilizado para treinar o modelo substituto. As imagens compostas de forma aleatória possibilitam que o método seja capaz de evitar algumas defesas propostas até o momento contra ataques de extração. Apesar dos bons resultados, sendo capaz de copiar 99% da acurácia dos modelos alvo, e de ter utilizado somente os rótulos únicos nos ataques, o método ainda é muito custoso, realizando milhões de chamadas a API do modelo alvo.

Já Chen et al. (2019) propôs um método diferente, chamado de DAFL (*Data-Free Learning*), que é capaz de gerar uma rede substituta através do uso de uma RGA para extrair as informações da rede alvo. O principal benefício do DAFL é que nenhum conjunto de dados de treino é necessário. O modelo alvo é utilizado como discriminador e o gerador é otimizado para gerar imagens sintéticas capazes de enganar o modelo alvo e depois gerar um conjunto para treinar o modelo substituto. O treinamento acontece através da repetição das seguintes etapas, até convergir: (*i*) o gerador produz uma quantidade de imagens sintéticas; (*ii*) as imagens são avaliadas pelo modelo alvo que fornece rótulos para cada imagem; (*iii*) os pesos do gerador são otimizados via *backpropagation*; (*iv*) o gerador produz novas imagens sintéticas; (*v*) as novas imagens são avaliadas pelo modelo alvo que fornece rótulos para cada imagem; (*vi*) o conjunto de dados composto pelas novas imagens e os rótulos fornecidos pelo modelo alvo é utilizado para treinar a rede cópia. O DAFL foi capaz de atingir 92.22% de acurácia em um conjunto de dados de teste do CIFAR-10 (KRIZHEVSKY; HINTON et al., 2009), superando outros métodos de extração de modelos.

Yuan et al. (2020) foi capaz de copiar modelos treinados nos principais conjuntos de dados de imagens, sem a necessidade de imagens do DP ou NDP. O método utiliza, de forma iterativa e repetitiva, um CDA formado por imagens sintéticas para extrair rótulos da rede alvo, produzir um modelo substituto capaz de aprimorar o CDA de imagens sintéticas a próxima iteração. Entretanto, apesar dos bons resultados apresentados pelo trabalho de Yuan et al. (2020) quanto pelo DAFL, ambos os trabalhos requerem uma quantidade enorme de consultas via API do modelo alvo e utilizam os *soft labels* como retorno do modelo alvo. Isso torna o método possível para distilar conhecimento, mas inviável como um ataque de extração contra modelo caixa preta, em cenários reais.

Os trabalhos de extração de modelos caixa preta propostos, até o momento, fazem uso de um grande número de requisições via API do modelo alvo, o que requer uma quantidade suficiente de imagens, sejam elas do domínio do problema (TRAMÈR et al.,

2016; SHI; SAGDUYU; GRUSHIN, 2017), imagens naturais aleatórias (CORREIA-SILVA et al., 2021; MOSAFI; DAVID; NETANYAHU, 2019b) ou sintéticas (CHEN et al., 2019). Imagens naturais aleatórias são facilmente encontradas na Internet e Correia-Silva et al. (2021) foi capaz de copiar redes com boa performance a partir de 100 mil requisições. Entretanto, um dos problemas que podem acontecer nessa abordagem é o modelo alvo atribuir o mesmo rótulo para todas, ou quase todas, imagens naturais, o que inviabiliza a extração ou requer que o atacante utilize um número maior de imagens do NDP e, consequentemente, mais requisições, para explorar as fronteiras de decisão do modelo alvo. Embora as imagens do DP sejam mais difíceis de encontrar e rotular, este trabalho propõem a construção de uma RGA para aumentar o número de imagens do DP, de forma sintética, e possibilitar a extração, a partir de um pequeno conjunto de imagens do DP.

4 Metodologia

Para analisar a possibilidade de ataques do tipo Copycat em Redes Neurais Convolucionais (RNCs) com imagens geradas à partir de Redes Generativas Adversárias (RGAs), tornam-se necessários os seguintes passos: (a) gerar modelos alvo para realizar os ataques, fornecendo um ponto de acesso para o atacante submeter imagens e obter os rótulos para extrair o conhecimento; (b) gerar um ataque Copycat para a comparação de resultados, fornecendo um *baseline* para comparar os demais modelos de ataque gerados; (c) treinar as RGAs para gerar novas imagens de ataque, a partir de um conjunto pequeno de imagens do Domínio do Problema (DP), e com isso, construir um conjunto de dados maior para o ataque; (d) efetuar o ataque Copycat com as imagens geradas pela RGA, para verificar se esse tipo de ataque é possível de ser realizado; e (e) efetuar a comparação com o ataque Copycat, onde os modelos alvo, os modelos substitutos da Copycat tradicional e os modelos substitutos da Copycat com imagens das RGAs serão avaliados em um conjunto de dados de teste e comparados entre si.

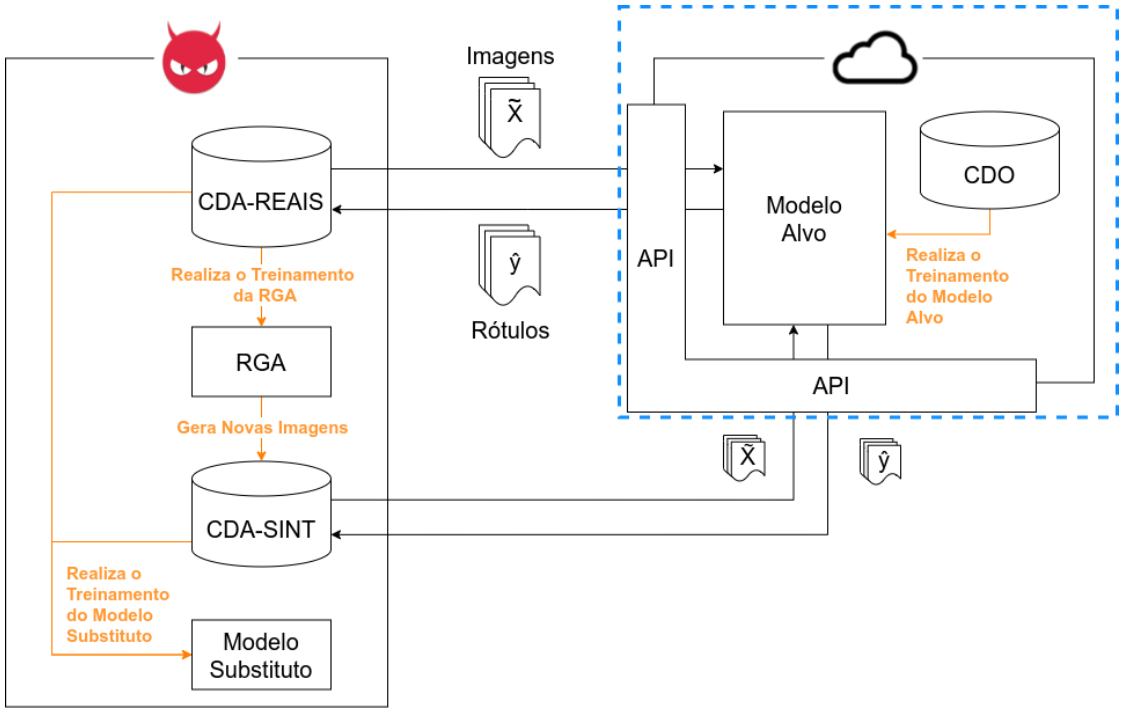

Figura 22 – Visão geral da metodologia proposta. Na direita (caixa azul pontilhada), o modelo alvo, treinado com o CDO confidencial, é servido publicamente via API, recebendo imagens como entrada e fornecendo os rótulos como saída. Na esquerda, o CDA-REAIS é composto de imagens reais do DP rotuladas pelo modelo alvo para treinar uma RGA. Um novo conjunto CDA-SINT, composto por imagens sintéticas do DP, produzidas pela RGA, e rótulos do modelo alvo, é utilizado para treinar o modelo substituto.

Na metodologia proposta, o modelo alvo é treinado para resolver algum problema específico (Figura 22, caixa azul pontilhada). Após o treinamento, seus parâmetros, sua arquitetura e seu Conjunto de Dados Original (CDO) são mantidos em segredo dentro da API, gerando assim o modelo alvo de caixa preta, que recebe somente as imagens de entrada e que fornece somente os rótulos (número inteiro indicando a classe da imagem) como saídas. Iniciando o método proposto (Figura 22, restante da imagem), o atacante faz a extração dos rótulos do modelo alvo (obtendo suas predições) para o conjunto de imagens do DP. Os rótulos extraídos do modelo alvo são denominados Rótulos Roubados (RR). Nessa etapa, mesmo que o atacante possua os rótulos dessas imagens, eles são descartados, pois deseja-se imitar o modelo alvo ao invés de criar um modelo competitivo. Então, o atacante gera o Conjunto de Dados de Ataque (CDA) inicial (CDA-REAIS) com as imagens reais do DP e os rótulos extraídos do modelo alvo. Esse conjunto é utilizado para o treinar a RGA, que após treinada, é utilizada para gerar imagens sintéticas. Como a RGA foi treinada por imagens do DP, o atacante espera que as imagens sintéticas sejam próximas ou que pertençam ao mesmo espaço do DP.

Continuando o ataque, ainda no intuito de imitar o modelo alvo, o atacante extrai os rótulos das imagens sintéticas do modelo alvo e gera um novo conjunto de ataque (CDA-SINT). Nesse ponto, o modelo substituto é treinado com o CDA-SINT e sua acurácia é medida. Testes também serão efetuados com um modelo substituto treinado com as imagens sintéticas e rótulos originais fornecidos pela RGA e outro treinado com a união do CDA-REAIS e do CDA-SINT. Para comparação, um ataque Copycat com imagens naturais de Não Domínio do Problema (NDP) também é realizado. A avaliação desses modelos é realizada em um conjunto de testes comum a todos os modelos, denominado Conjunto de Dados de Validação (CDV). Por fim, a metodologia proposta foi aplicada em três cenários diferentes, com três problemas investigados, sendo eles: Classificação de Dígitos Manuscritos (CDM), Classificação de Objetos Gerais (COG) e Classificação de Número da Casa Visto da Rua (CNCVR). Esses problemas foram escolhidos porque os conjuntos de imagens de domínio público utilizados em cada problema (descritos na seção 4.1) possuem imagens pequenas, com tamanhos iguais ou inferiores a 32×32 *pixels*. Isso permite que o custo computacional necessário não ultrapasse o limite disponível para este trabalho.

Em um cenário real, existe um custo associado a cada requisição feita pelo atacante ao modelo alvo. Por isso, este trabalho estabelece a restrição de que no máximo 100 mil requisições sejam feitas a API, com o objetivo de garantir que o custo relacionado a aplicação da metodologia, em um cenário real, não cresça a ponto de tornar o método inviável. Além disso, o número de 100 mil requisições foi escolhido devido ao Correia-Silva et al. (2021) ter demonstrado que o método Copycat é capaz de copiar modelos alvo a partir dessa quantidade de requisições e devido ao recurso computacional disponível. O restante deste Capítulo apresenta com mais detalhes a metodologia proposta e os passos

necessários para realização dos objetivos deste trabalho, com detalhes da implementação, configurações gerais dos modelos e os recursos computacionais utilizados.

4.1 Modelos Alvo para Realização dos Ataques

A rede f é um modelo treinado com um CDO para resolver cada um dos problemas de investigados. O CDO é composto por $\{(x_i, y_i)\}_{i=1}^{N}$, que é o conjunto de N imagens x_i com os respectivos rótulos y_i. Devido a natureza das RNPs, seu treinamento requer um número considerável de imagens, o que torna o custo para obter CDO e, consequentemente, o modelo treinado f, elevado (GOODFELLOW et al., 2016). O modelo f é parametrizado por θ, que representa seus parâmetros e conhecimento do modelo, e sua saída é dada pela Equação 4.1, onde x_i é uma imagem de entrada e como o tipo de ataque é $\hat{y}_i$ é um valor inteiro que representa a qual das $k \in K$ classes do problema x_i pertence (CORREIA-SILVA et al., 2021).

$$\hat{y}_i = f_\theta(x_i) \tag{4.1}$$

O treinamento de f com o CDO visa encontrar o melhor conjunto de parâmetros θ, através das iterações via *backpropagation* para minimizar a Equação 4.2, onde θ^* representa o conjunto de parâmetros inicial e $\mathcal{L}$ é a função de entropia cruzada. Durante o treinamento de f, a função $\mathcal{L}$ também é calculada para um conjunto de dados separado (CDV), para avaliar a capacidade de generalização de f. Assim como o CDO, o CDV, é composto por imagens do conjunto original, mas sem nenhuma imagem pertencente ao CDO (i.e., CDV $\cap$ CDO $= \{\}$).

$$f_\theta = \arg\min_{\theta^*} \mathcal{L}(f_{\theta^*}(x_i), y_i) \tag{4.2}$$

O modelo f é denominado modelo alvo porque é a rede ao qual o atacante deseja extrair informações, com o objetivo potencial de construir uma rede substituta $\hat{f}$ que se comporta de forma semelhante a f (RIGAKI; GARCIA, 2020). Do ponto de vista do atacante, a arquitetura de f, os parâmetros θ, o CDO e o CDV, são desconhecidos. O modelo f será disponibilizado de forma similar a uma API, fornecendo acesso para o atacante submeter imagens e obter rótulos para extrair conhecimento. Por se tratar de ataques do tipo Copycat (CORREIA-SILVA et al., 2021), f retornará somente o valor unitário que representa a qual classe cada imagem pertence, conforme Equação 4.1.

Para a arquitetura de f, diversas RNCs podem ser utilizadas, desde uma arquitetura própria, desenhada pelos donos do modelo alvo, até arquiteturas disponíveis na literatura, como a VGG (SIMONYAN; ZISSERMAN, 2014), a ResNet (HE et al., 2016), a AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2017), a LeNet SENet (HU; SHEN; SUN,

2018), entre outras. Essas arquiteturas podem ser encontradas na Internet já pré treinadas em largas bases populares, como, por exemplo, o ImageNet (DENG et al., 2009). Embora arquiteturas mais complexas apresentem uma boa acurácia em classificação de imagens, devido a limitação de recurso computacional disponível e ao bom desempenho já obtido na VGG16 (LIU; DENG, 2015), esta arquitetura, pré treinada no ImageNet, foi adotada para a construção dos modelos alvo. Por fim, a Tabela 1 descreve quais conjuntos de dados de domínio público foram utilizados na construção do CDO e do CDV para o treino dos modelo alvo de cada problema investigado. A quantidade de imagens utilizadas em cada conjunto seguiu a divisão padrão recomendada pelos autores. Esses conjuntos foram utilizados devido a restrições de recursos computacionais e as imagens que os compõem foram, quando necessárias, redimensionadas para o tamanho 32×32 *pixels* para reduzir a complexidade computacional dos problemas, mas sem afetar as acurácias dos modelos alvo.

Tabela 1 – Detalhes dos problemas investigados, bases de domínio público utilizadas e quantidade de imagens em cada conjunto de dados.

Problemas	Conjuntos de Dados	# CDO	# CDV
CDM	MNIST (LECUN et al., 1998)	60.000	10.000
COG	CIFAR-10 (KRIZHEVSKY; HINTON et al., 2009)	50.000	10.000
CNCVR	SVHN (NETZER et al., 2011)	73.257	26.032

4.2 Ataque Copycat para Comparação dos Resultados

Para executar o ataque Copycat, o atacante precisa de uma quantidade de imagens, para realizar a extração de conhecimento dos modelos alvo. Essas imagens podem ser pertencentes ao DP ou imagens naturais (NDP) (CORREIA-SILVA et al., 2021). Durante a extração, o atacante envia essas imagens ($\tilde{x}_i$) para a API do modelo alvo f, que retorna valores que representam a qual classe cada imagem pertence. Após a extração, o atacante possui um novo conjunto de dados $\{(\tilde{x}_i, \hat{y}_i)\}_{i=1}^{M}$, onde $\hat{y}_i = f(\tilde{x}_i)$ são os rótulos roubados do modelo alvo. Esse novo conjunto, é conhecido como CDA-REAIS (Figura 22).

Neste trabalho, o CDA-REAIS para o ataque Copycat (CORREIA-SILVA et al., 2021) foi construído através da seleção aleatória de 100 mil imagens naturais do ImageNet, redimensionadas para o tamanho 32×32 *pixels*, e rotuladas roubados de f. O número de imagens escolhido visa respeitar o limite de 100 mil requisições, previamente estabelecido. Note que o CDA-REAIS pode possuir um desbalanceamento do número de imagens $\tilde{x}_i$ para cada classe, porque o modelo alvo pode classificar mais imagens para uma classe ou outra.

Por fim, o CDA-REAIS é utilizado para treinar um modelo substituto $\hat{f}$ e seu treinamento pode ser representado conforme a Equação 4.3, onde ϕ representa um conjunto diferente de parâmetros. Os modelos substitutos obtidos através da aplicação do ataque Copycat tradicional, com imagens do NDP, serão utilizados como *baseline* para comparação com os modelos substitutos obtidos através da metodologia proposta (i.e. ataque Copycat com imagens geradas por RGA).

$$\hat{f}_{\phi} = \arg\min_{\phi*} \mathcal{L}(\hat{f}_{\phi*}(\tilde{x}_i), \hat{y}_i) \tag{4.3}$$

Conforme apontado por Correia-Silva et al. (2021), em cenários reais, o atacante não tem conhecimento da arquitetura do modelo alvo f ou pode inferi-lá a partir de outros tipos de ataque disponíveis na literatura. O atacante também pode escolher uma arquitetura diferente $\hat{g}$ devido a restrições relacionadas ao recurso computacional disponível. Entretanto, este trabalho utiliza a mesma arquitetura para o modelo alvo e o modelo substituto, ou seja, a VGG16 pré treinada no ImageNet. Uma vez que o objetivo deste trabalho é analisar a viabilidade do ataque Copycat com imagens sintéticas geradas a partir de RGA e devido ao trabalho original da Copycat utilizar a mesma arquitetura para o modelo alvo e o modelo substituto.

4.3 Treinamento da RGA para Gerar Imagens Sintéticas

O atacante pode criar, gerar ou obter imagens do DP a partir do conhecimento do problema que o modelo alvo foi treinado e do contexto das classes que o modelo alvo é capaz de classificar (CORREIA-SILVA et al., 2018). Entretanto, imagens do DP são mais escassas que imagens do NDP. Com o objetivo de imitar o modelo alvo, o atacante pode utilizar essas imagens do DP e os rótulos roubados do modelo alvo, através da extração de conhecimento, para compor o CDA-REAIS. Uma RGA é treinada a partir do CDA-REAIS (Figura 22) com imagens do DP, para produzir imagens sintéticas como intuito de aumentar o número de imagens similares ao DP disponíveis.

Dentre as diversas arquiteturas de RGAs presentes na literatura, testes preliminares foram realizados com a Vanilla GAN (GOODFELLOW et al., 2014), a Conditional GAN (MIRZA; OSINDERO, 2014) e a Auxiliary Classifier GAN (ODENA; OLAH; SHLENS, 2017). A Conditional GAN apresentou melhores resultados e tempo de execução, sendo a arquitetura escolhida para utilizar nesse trabalho. Além disso, o uso da arquitetura Conditional GAN permite que a RGA seja capaz de receber o rótulo desejado junto com o vetor de valores aleatórios, condicionando que a imagem sintética gerada pertença a alguma das classes do problema. Esses rótulos serão referenciados como rótulos originais (RO). Como padrão, as RGAs foram treinadas por 500 épocas, com Taxa de Aprendizado igual a 5×10^{-4} e política de descida do gradiente.

Tabela 2 – Detalhes dos conjuntos de imagens do domínio de problema utilizados para cada problema investigado.

Problemas	Conjuntos de Dados
CDM	EMNIST (COHEN et al., 2017)
COG	STL-10 (COATES; NG; LEE, 2011) e ImageNet (DENG et al., 2009)
CNCVR	SVHN (NETZER et al., 2011)

A Tabela 2 exibe os conjuntos de imagens de domínio público utilizados na extração de conhecimento para construir o CDA-REAIS com imagens de DP. Foi definida como restrição o número de mil imagens de cada classe para cada problema específico. O CDA-REAIS de todos os problemas consumiu 10 mil requisições a API. Para a tarefa de CDM, 1.000 imagens de cada classe foram aleatoriamente selecionada do EMNIST (COHEN et al., 2017), formando o CDA-REAIS de 10.000 imagens. Já para tarefa de COG, como o modelo alvo foi treinado com o CIFAR-10 (KRIZHEVSKY; HINTON et al., 2009), dois conjuntos foram utilizados para compor o CDA-REAIS. Do STL-10 (COATES; NG; LEE, 2011) foram utilizadas 1.000 imagens aleatórias para cada uma das 9 classes das quais ele possui interseção com o CIFAR-10 e foram aleatoriamente selecionadas 1.000 imagens do ImageNet (DENG et al., 2009) para classe sapo. Embora o SVHN (NETZER et al., 2011) seja o mesmo conjunto utilizado para treinar o modelo alvo para a tarefa de CNCVR, para a construção do CDA-REAIS, 1.000 imagens de cada classe foram selecionadas de forma aleatória a partir do conjunto de imagens extras do SVHN, garantindo a restrição de não intersecção entre o CDA-REAIS e o conjuntos CDO e CDV.

Para a avaliação da RGA, durante seu treinamento foram coletados dados para calcular as seguintes métricas: (a) a acurácia do modelo alvo f nas imagens sintéticas, para avaliar se os rótulos originais, condicionados a geração das imagens sintéticas, são iguais aos rótulos roubados do modelo alvo; (b) o FID entre as imagens sintéticas e as imagens do CDO, para verificar se as imagens sintéticas são espacialmente próximas ao domínio do problema; e (c) o FID entre as imagens sintéticas e as imagens do CDA-REAIS, para verificar se as imagens sintéticas são espacialmente próximas as imagens de DP obtidas pelo atacante e rotuladas pelo modelo alvo, definidas como imagens reais (não falsas) para o Discriminador da RGA. Para o cálculo das métricas, foi utilizado um conjunto de 100 imagens sintéticas ($G(Z)$), geradas a cada 100 épocas do treinamento da RGA a partir de um conjunto de vetores de entrada aleatório pré definido (Z), que é utilizado repetitivamente de forma fixa, para que as métricas correspondam sempre as "mesmas" imagens sintéticas. Esse conjunto será enviado como requisição ao modelo alvo para calcular a primeira métrica. Já as últimas duas métricas serão calculadas localmente através da *Inception Network*. Como essa informação é usada somente para avaliação do trabalho e não altera o treinamento da RGA de nenhuma forma, as requisições ao modelo alvo não

serão contadas dentro da restrição de 100 mil requisições.

Após o treinamento, G deve ser capaz de gerar novas imagens sintéticas que seguem aproximadamente a distribuição do CDA-REAIS e apresentem uma variedade entre as classes, para serem utilizadas depois para atacar f com mais eficiência, ou seja, explorando melhor o espaço de classificação. Dessa forma, para evitar o *mode collapse* de G, o Mode Seeking (MAO et al., 2019) foi utilizado, pela simplicidade e por ter apresentado os melhores resultados da literatura, até o momento.

Além da metodologia proposta, que propõem que um pequeno conjunto de imagens de DP seja utilizado pelo atacante para compor o CDA-REAIS. Um experimento adicional foi produzido utilizando um pequeno conjunto de imagens de NDP para compor o CDA-REAIS. Esse experimento visa avaliar a diferença entre o ataque Copycat com imagens sintéticas similares ao DP e imagens sintéticas similares ao NDP e os impactos no mapeamento do espaço de classificação do modelo alvo. Para esse experimento, as RGAs foram treinadas com o CDA-REAIS composto de 10 mil imagens naturais retiradas de forma aleatória do ImageNet (DENG et al., 2009) e rótulos roubados do modelo alvo.

4.4 Ataque Copycat com Imagens Geradas pela RGA

Com a RGA treinada com a arquitetura Conditional GAN (MIRZA; OSINDERO, 2014), o Gerador pode ser utilizado para gerar imagens sintéticas para rótulos específicos, condicionados a valores do vetor de entrada. Dessa forma, o Gerador será utilizado para gerar 90 mil imagens sintéticas, sendo 9 mil imagens para cada classe. Os Rótulos Originais (RO) de cada uma dessas imagens indicam os rótulos que foram condicionados a geração de cada imagem. Essas imagens sintéticas serão utilizadas pelo atacante para extrair o conhecimento do modelo alvo, obtendo os rótulos roubados para cada imagem sintética e completando assim o limite de 100 mil requisições na API. O par imagem sintética e rótulo roubado irá compor o CDA-SINT (Figura 22).

Nesse momento do ataque, o atacante possui dois conjuntos, o CDA-REAIS que possui 10 mil imagens inicialmente obtidas e rótulos roubados do modelo alvo, e o CDA-SINT que possui 90 mil imagens sintéticas produzidas pela RGA, também com rótulos roubados do modelo alvo. Durante os experimentos, três modelos substitutos serão treinados com três conjuntos diferentes:

1. **SINT-RR**: Representa o primeiro modelo substituto, que será treinado com o CDA-S.

2. **SINT-RO**: Representa o segundo modelo substituto, que será treinado com o CDA-S, porém com os rótulos originais das imagens sintéticas, para avaliar a qualidade dos rótulos originais.

3. **SINT+REAIS**: Representa o terceiro modelo substituto, que será treinado a união do CDA e do CDA-S.

4.5 Comparação dos Resultados

O CDV foi utilizado nesse trabalho para avaliar a equivalência entre $\hat{f}$ e f, verificando assim se a extração foi realizada com sucesso. Dessa forma, a acurácia calculada no CDV, servirá de métrica para avaliar e comparar os modelos substitutos de *baseline* obtidos a partir do ataque Copycat tradicional e os modelos substitutos obtidos a partir do ataque Copycat com imagens sintéticas. A acurácia representa a quantidade percentual de imagens que foram corretamente classificadas pelo modelo, divididas pelo total de imagens do CDV. De acordo com Correia-Silva et al. (2021), a cópia é reconhecida como perfeita se a rede atingir uma acurácia idêntica ao modelo alvo no CDV.

4.6 Implementação e Configurações

A implementação do projeto foi baseada na linguagem de programação Python, que é amplamente utilizada em todo o mundo, principalmente para implementar algoritmos de aprendizado de máquina. A RGA e as RNPs para o modelo alvo e para o modelo cópia foram implementadas em *PyTorch* (PASZKE et al., 2019), biblioteca de código aberto para desenvolvimento de redes neurais, escrita em Python e desenvolvida pelo *Facebook*, utilizada na implementação dos principais trabalhos atualmente (MAO et al., 2019; OREKONDY; SCHIELE; FRITZ, 2019). O *PyTorch* fornece uma maneira conveniente de definir e treinar qualquer tipo de RNA. Todos os códigos relacionados a implementação dos experimentos estão divulgados publicamente [1].

O ataque Copycat, disponibilizado publicamente, foi originalmente implementado em Python com a biblioteca Caffe (JIA et al., 2014). Para ser possível comparar os resultados da Copycat com os resultados deste trabalho, o ataque foi reimplementado com o PyTorch (PASZKE et al., 2019).

Apesar de todos os conjuntos de dados, necessários na realização deste trabalho, estarem disponíveis publicamente na Internet, durante os experimentos para cada um dos problemas citados, os conjuntos de dados necessários para construção do CDO e CDV, assim como os conjuntos necessários para imagens de DP do atacante, foram construídos através da API do *PyTorch* (PASZKE et al., 2019), que fornece os conjuntos de dados via de objetos Python. Todos os modelos alvos e substitutos foram treinados por 50 iterações, escolhidas através de experimentos preliminares, com a arquitetura VGG16 e utilizando o Stochastic Gradient Descent (SGD) com Taxa de Aprendizado igual a 10^{-3} e política de

[1] https://github.com/brendalf/gan-attack

descida gradual.

4.7 Recursos Computacionais

Todo o desenvolvimento do projeto e execução dos experimentos foi realizado em computador pessoal, com a seguinte configuração: (i) sistema operacional Ubuntu 20.04.2 LTS, com *kernel* Linux v5.8.18; (ii) processador Intel(R) Core $i7 - 8700K$ CPU, 3.70GHz com 12 núcleos físicos; (iii) memória RAM de 32 GB; (iv) unidade de armazenamento de 512 GB (SSD); (v) placa de vídeo NVIDIA GTX 1080, com 8 GB de memória dedicada.

5 Resultados

Neste capítulo são apresentados os resultados obtidos através da metodologia proposta. Inicialmente, serão apresentados os resultados da análise efetuada do treinamento das RGAs, onde uma avaliação qualitativa de desempenho foi realizada através da FID e da acurácia do modelo alvo sobre amostras sintéticas. Em seguida, serão apresentados os resultados de todos os modelos treinados, substitutos e alvo. Por fim, uma análise de todos os resultados será realizada.

5.1 Análise do Treinamento da RGA e das Imagens Sintéticas

Para cada problema investigado, duas RGAs foram treinadas, uma utilizando o CDA-REAIS com imagens do DP e outra utilizando o CDA-REAIS com imagens do NDP. Por isso, as RGAs serão representadas como RGA-DP quando referidos aos experimentos com imagens do DP, e RGA-NDP quando referidos aos experimentos com imagens do NDP. A Figura 23 exibe o desempenho do treinamento da RGA-DP (imagens superiores) e da RGA-NDP (imagens inferiores) através do FID e da acurácia do modelo alvo sobre as imagens sintéticas, geradas a cada 100 épocas do treinamento.

Como o FID representa a distância espacial entre as imagens e quanto menor melhor, pode-se observar que a RGA-DP é capaz de aproximar espacialmente as imagens sintéticas das imagens do CDA-REAIS (Figura 23, *b*), para todos os problemas, atingindo um *plateau* após 100 épocas de treinamento. Pode-se observar que as imagens sintéticas também se aproximam das imagens do CDO (Figura 23, *a*). A RGA-NDP parece também ser capaz de aproximar as imagens sintéticas das imagens do CDA-REAIS (Figura 23, *e*), porém com valores de FID maiores do que a RGA-DP. Isso pode ser um reflexo da complexidade das imagens naturais de diferentes classes retiradas de forma aleatória do ImageNet (DENG et al., 2009). Em relação ao FID para imagens do CDO (Figura 23, *d*), a RGA-NDP não é capaz de diminuir o FID através das épocas do treinamento.

A acurácia do modelo alvo sobre as imagens sintéticas representa o quanto os rótulos originais, produzidos pelas RGAs, são iguais ao rótulos roubados fornecidos pelos respectivos modelos alvo de cada problema. Pode-se notar que as acurácias das imagens sintéticas produzidas por RGA-DP (Figura 23, *c*) são maiores, em geral, que as acurácias das imagens sintéticas produzidas por RGA-NDP (Figura 23, *f*). Após o treinamento das RGAs, a acurácia da RGA-DP para o problema COG foi de $46,8\%$ contra $19,4\%$ pela RGA-NDP do mesmo problema. Para o problema CNCVR, as acurácias foram de 60% para a RGA-DP contra 14% da RGA-NDP. Por fim, para o problema de CDM, a RGA-DP apresentou uma acurácia de 21% contra 15% da RGA-NDP, sendo o pior resultado entre

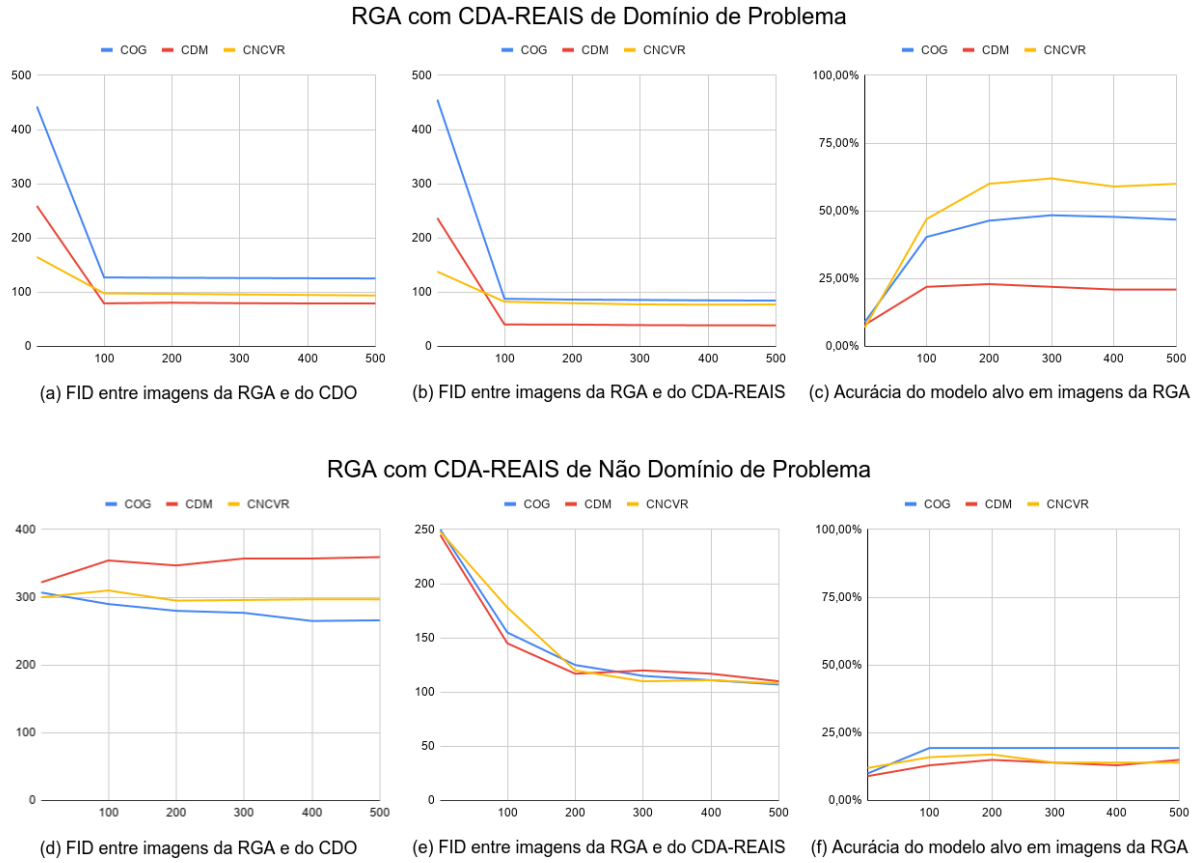

Figura 23 – Desempenho durante o treinamento das RGAs. As linhas azuis, vermelhas e amarelas representam os problemas COG, CDM e CNCVR, respectivamente. As imagens superiores informam os resultados da RGA treinada com CDA-REAIS com imagens de DP e as imagens inferiores informam os resultados da RGA treinada com CDA-REAIS com imagens de NDP. As imagens a e d representam a evolução do FID entre as imagens sintéticas e as imagens do CDO. As imagens b e e representam a evolução entre as imagens sintéticas e as imagens do CDA-REAIS utilizadas no treinamento da RGA. Por fim, as imagens c e f representam a acurácia do modelo alvo nas imagens sintéticas produzidas pelas RGAs

as RGA-DP, apesar de o FID entre as imagens sintéticas e as imagens do CDO terem apresentado o melhor resultado. Isso pode indicar que as imagens sintéticas são similares as imagens de DP contidas dentro do CDA-REAIS utilizado no treinamento da RGA-DP, porém os rótulos originais não são condizentes com as imagens sintéticas geradas.

A Figura 24 apresenta algumas imagens sintéticas, geradas pelas RGA-DP já treinadas, com os rótulos originais condicionados na geração das imagens. Essas imagens sintéticas foram usadas na extração de conhecimento para compor o CDA-SINT com os rótulos roubados. Em geral, pode-se notar que, assim como apresentado pelo FID, as imagens são semelhantes ao DP. Contudo, as imagens parecem não ser tão distintas para o problema COG, o que pode indicar que as classes são muito complexas para a RGA utilizada. Já para o problema de CDM, a RGA parece ser capaz de gerar imagens do DP

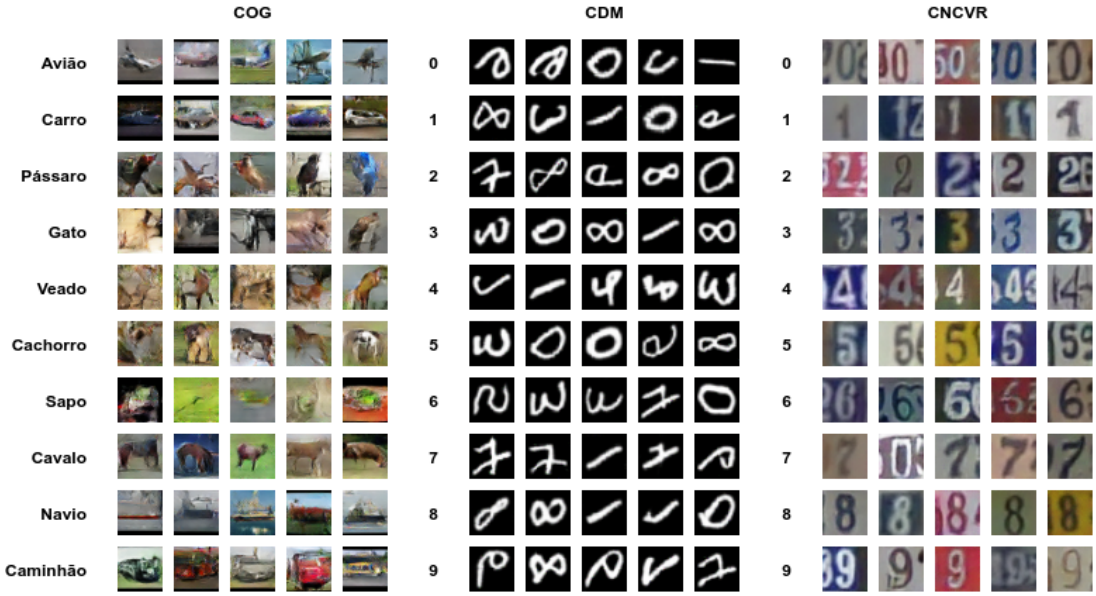

Figura 24 – Imagens sintéticas produzidas pela RGA-DP. Os rótulos apresentados são os rótulos originais condicionados na geração das imagens sintéticas. Essas imagens foram utilizadas para extrair conhecimento e compor o respectivo CDA-SINT. A coluna da esquerda apresenta as imagens geradas para o problema COG. A coluna central apresenta as imagens geradas para o problema CDM. Por fim, a coluna da direita apresenta as imagens geradas para o problema CNCVR.

com alta fidelidade, conforme indicado pelos valores obtidos com o FID (Figura 23, *b*). Entretanto, parece que a RGA não foi capaz de gerar imagens sintéticas para os rótulos originais corretos, como pode-se perceber, por exemplo, para a classe 2, que possui imagens que parecem mais da classe 7, 8 ou 9. Isso implica em baixa acurácia para o problema (Figura 23, *c*). Por fim, as imagens do CNCVR apresentam uma boa qualidade e confiança de classes, indicado pela acurácia de 60% obtidas no modelo alvo.

Da mesma forma, a Figura 25 apresenta algumas imagens sintéticas, geradas pelas RGA-NDP já treinadas, com os rótulos originais condicionados na geração das imagens. Essas imagens foram utilizadas na extração de conhecimento para compor o CDA-SINT com os rótulos roubados. Ao comparar as imagens geradas pelas RGA-NDP (Figura 25) com as imagens geradas pelas RGA-DP (Figura 24), pode-se notar que as RGA-NDP produzem imagens mais abstratas. Isso pode ocorrer porque as imagens pertencentes ao CDA-REAIS e definidas como verdadeiras para os Discriminadores das RGA-NDP são muito distintas entre si, uma vez que foram selecionadas de forma aleatória do ImageNet que possui mais de mil classes diferentes. Já as RGA-DP produzem imagens, conforme esperado, mais coerentes ao domínio do problema do modelo alvo.

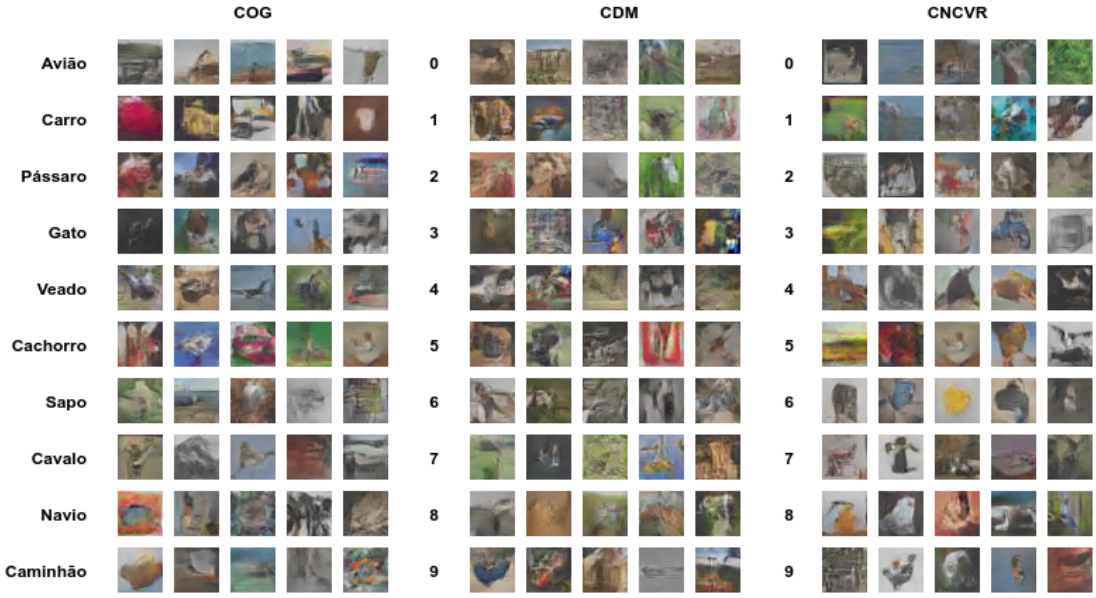

Figura 25 – Imagens sintéticas produzidas pela RGA-NDP. Os rótulos apresentados são os rótulos originais condicionados na geração das imagens sintéticas. Essas imagens foram utilizadas para extrair conhecimento e compor o respectivo CDA-SINT. A coluna da esquerda apresentam as imagens geradas para o problema COG. A coluna central apresentam as imagens geradas para o problema CDM. Por fim, a coluna da direita apresenta as imagens geradas para o problema CNCVR.

5.2 Resultados dos Modelos Gerados

Conforme descrito na metodologia, três modelos substitutos foram treinados para servir como *baseline* para comparação dos resultados, sendo eles: (Alvo) o próprio modelo alvo, (COMP) um modelo de comparação treinado com as imagens do DP utilizadas no ataque e rotuladas pelo próprio atacante e a (CC) Copycat (CORREIA-SILVA et al., 2021) com 100 mil imagens naturais do NDP. A Tabela 3 apresenta a acurácia de cada modelo obtida no CDV do respectivo problema.

Tabela 3 – Resultado dos modelos de *baseline*. A coluna COMP representa o modelo de comparação treinado com as imagens do DP utilizadas no ataque, porém com os rótulos anotados pelo próprio atacante. A coluna CC representa o ataque Copycat com imagens naturais aleatórias.

Problema	Alvo	COMP	CC
CDM	$98,90\%$	$44,68\%$	**94,04%**
COG	$91,86\%$	$62,01\%$	**85,97%**
CNCVR	$94,74\%$	**86,15%**	**85,48%**

Os modelos alvo atingiram resultados acima de 90% para todos os problemas, demonstrando boa capacidade de generalização, sendo que a acurácia mais baixa foi de

91,86% no problema de COG, que é o conjunto de dados mais complexo. Os ataques Copycat (CC) foram capazes de copiar os modelos alvo com desempenho esperado, atingindo uma acurácia de 94% para o CDM e 85% para os outros problemas. Já os modelos de comparação treinados com as imagens do DP com rótulos anotados pelo próprio atacante (COMP), atingiram uma acurácia mais alta para o problema de CNCVR e uma acurácia mais baixa para os outros dois. Essa acurácia mais baixa pode indicar que as imagens selecionadas para o ataque, apesar de pertencerem ao DP, não representam de forma adequada o espaço de classificação dos modelos alvo.

Em seguida, serão apresentados os resultados do ataque Copycat com imagens sintéticas geradas a partir de RGA, começando pela RGA-DP e depois a RGA-NDP. O prefixo D ou N será adicionado as siglas dos modelos substitutos obtidos através dos ataques com imagens sintéticas, para indicar modelos substitutos treinado a partir da RGA-DP e a RGA-NDP, respectivamente.

Tabela 4 – Resultado do ataque Copycat com imagens sintéticas de DP. A coluna DSINT-RO representa o modelo treinado com as imagens sintéticas do CDA-SINT com os rótulos originais, condicionados pela RGA-DP. A coluna DSINT-RR representa o modelo treinado com o CDA-SINT. A coluna DSINT+REAIS representa o modelo treinado com a união do CDA-SINT com o CDA-REAIS. Por fim, as colunas COMP e CC foram repetidas da Tabela 3 para facilitar a visualização dos resultados.

Problema	Alvo	COMP	CC	DSINT-RO	DSINT-RR	DSINT+REAIS
CDM	98,90%	44,68%	**94,04%**	14,98%	82,78%	82,68%
COG	91,86%	62,01%	**85,97%**	38,13%	84,14%	**85,53%**
CNCVR	94,74%	86,15%	85,48%	72,83%	**91,38%**	**91,76%**

A Tabela 4 exibe a acurácia dos modelos substitutos treinados a partir do RGA-DP. O primeiro modelo (representado por DSINT-RO), treinado com as imagens do CDA-SINT com os rótulos originais condicionados da RGA-DP, obteve uma baixa acurácia, demonstrando que a RGA-DP não foi capaz de gerar imagens nos rótulos corretos (rótulos originais diferentes dos rótulos roubados). O resultado desse modelos parecem estar diretamente relacionados a qualidade das imagens do DP utilizadas e o resultado do COMP. Quando as imagens do DP não são representativas, o modelo alvo as classifica de forma errada, agrupando imagens de classes distintas em classes iguais. Isso faz com que a RGA-DP receba imagens que pertencem a mesma classe como se fossem classes diferentes, o que confunde o Gerador e o Discriminador, como por exemplo, as imagens produzidas para o problema CDM na Figura 24. Os modelos treinados com o CDA-SINT (representados por DSINT-RR) atingiram acurácias acima de 80% para todos os problemas, com destaque para o problema de CNCVR, que atingiu 91,38%. Por fim, pode-se notar que os modelos treinados com a união do CDA-SINT e do CDA-REAIS (representado por

DSINT+REAIS), obtêm os melhores resultados dos três tipos avaliados, com acurácia similar ao DSINT-RR no CDM e superando o Copycat (CC) no CNCVR.

Tabela 5 – Resultado do ataque Copycat com imagens sintéticas de NDP. A coluna NSINT-RO representa o modelo treinado com as imagens do CDA-SINT com os rótulos originais, condicionados pela RGA-NDP. A coluna NSINT-RR representa o modelo treinado com o CDA-SINT. A coluna NSINT+REAIS representa o modelo treinado com a união do CDA-SINT com o CDA-REAIS. Por fim, a coluna CC foi repetida da Tabela 3 para facilitar a visualização dos resultados.

Problema	Alvo	CC	NSINT-RO	NSINT-RR	NSINT+REAIS
CDM	98,90%	**94,04%**	7,50%	83,70%	91,03%
COG	91,86%	**85,97%**	18,47%	80,47%	**85,02%**
CNCVR	94,74%	**85,48%**	14,69%	83,98%	**85,75%**

A Tabela 5 apresenta a acurácia dos modelos substitutos treinados a partir do RGA-NDP. O primeiro modelo (representado por NSINT-RO), treinado com as imagens do CDA-SINT com os rótulos originais condicionados da RGA-NDP, obteve uma acurácia muito baixa, o que pode indicar que o número de imagens utilizados para compor o CDA-REAIS e treinar a RGA-NDP não foi capaz de capturar o espaço de classificação dos modelos alvo de forma suficiente. Os modelos treinados com o CDA-SINT (representados por NSINT-RR) atingiram acurácias acima de 80% para todos os problemas. Por fim, pode-se notar que os modelos treinados com a união do CDA-SINT e do CDA-REAIS (representado por NSINT+REAIS), obtêm os melhores resultados dos três tipos avaliados, sendo capaz de atingir uma performance similar a Copycat (CC) para os problemas de COG e de CNCVR.

Tabela 6 – Visão geral dos principais resultados.

Problema	Alvo	COMP	CC	DSINT+REAIS	NSINT+REAIS
CDM	98,90%	44,68%	**94,04%**	82,68%	91,03%
COG	91,86%	62,01%	**85,97%**	**85,53%**	**85,02%**
CNCVR	94,74%	86,15%	85,48%	**91,76%**	85,75%

Todos os modelos treinados com imagens sintéticas (DP ou NDP) com rótulos originais condicionados, apresentaram resultados inferiores ao modelos treinados com os rótulos extraídos. O que pode indicar que a quantidade de imagens com rótulos do modelo alvo, utilizadas para treinar as RGAs, não foram suficientes para mapear o espaço de classificação. Outro ponto importante é que os modelos treinados com a união dos conjuntos foi superior aos modelos treinados somente com imagens sintéticas. Dessa forma, para facilitar a discussão final, a Tabela 6 sumariza os principais modelos gerados, destacando

em negrito o modelo com melhor acurácia para cada problema investigado. Embora a necessidade de imagens do DP, os ataques Copycat com imagens sintéticas do DP conseguiram acurácias superiores aos ataques com imagens sintéticas do NDP e similares aos ataques Copycat tradicionais (CC), exceto para o problema CDM, devido a baixa representatividade das imagens do DP. As imagens produzidas pela RGA-DP (Figura 24) parecem, aos olhos humanos, mais relacionadas a tarefa de classificação que as imagens abstratas produzida pela RGA-NDP (Figura 25). Entretanto, essas imagens também foram capazes de copiar os estados da rede não visualizados pelos olhos humanos e produzir modelos substitutos eficientes para a imitação. Por fim, esse método, utilizando 10 mil naturais do ImageNet (DENG et al., 2009) e 90 mil imagens sintéticas, atingiu acurácias parecidas com o ataque Copycat tradicional, que utilizou 100 mil imagens naturais.

6 Conclusão

Este trabalho propôs a realização de um ataque Copycat em RNCs caixa preta com imagens sintéticas produzidas por RGAs, treinadas a partir de um pequeno conjunto de imagens do DP obtidas pelo atacante. Após a análise dos resultados, pode-se verificar que as imagens sintéticas produzidas pelas RGAs são visualmente similares às imagens do DP e que os modelos substitutos obtiveram resultados satisfatórios. Por isso, concluímos que é possível utilizar RGAs para produzir novas imagens sintéticas à partir de imagens do DP e utilizá-las para realizar ataques Copycat (CORREIA-SILVA et al., 2021) contra modelos de RNCs caixa preta, uma vez que, ainda com imprecisões, essas imagens sintéticas ajudam a mapear o espaço de classificação dos modelos alvo.

Como obter imagens do DP por parte do atacante é uma tarefa custosa, principalmente quando o atacante só tem conhecimento sobre o contexto do problema que o modelo alvo é capaz de resolver, o uso de uma RGA pode ser uma solução para aumentar o conjunto de ataque. Entretanto, os resultados obtidos para o problema de CDM levanta o questionamento de que a qualidade do modelo substituto e, consequentemente, do ataque, parecem estar diretamente relacionados ao quanto as imagens do DP utilizadas na extração inicial de rótulos se assemelham ao espaço dimensional do modelo alvo.

Além do uso de imagens do DP, as análises demonstraram que o método também pode ser utilizado a partir de um conjunto menor de imagens de NDP. Embora os resultados apresentados sejam próximos aos resultados dos ataques Copycat tradicional, como as imagens naturais estão disponíveis em abundância na Internet, acredita-se não ser viável utilizar RGAs para produzir imagens sintéticas de NDP, devido ao alto custo computacional necessário para treinar um RGA.

Como extensão e melhorias ao trabalho apresentado pode-se recomendar: a análise da metodologia para outros conjuntos de dados mais complexos; a análise do impacto da utilização de arquiteturas de redes pré treinadas no ImageNet (DENG et al., 2009) como modelo alvo; a análise da metodologia proposta contra os métodos de defesa contra ataque de privacidade; e a análise da metodologia proposta com outras arquiteturas de RGA e o uso de diferentes quantidades de imagens do DP.

Referências

BASART, S. Analysis of generative adversarial models. *THE UNIVERSITY OF CHICAGO*, 2017. Citado 4 vezes nas páginas 31, 32, 33 e 34.

BENGIO, Y. et al. Deep generative stochastic networks trainable by backprop. In: PMLR. *International Conference on Machine Learning*. [S.l.], 2014. p. 226–234. Citado na página 30.

BJORCK, J. et al. Understanding batch normalization. *arXiv preprint arXiv:1806.02375*, 2018. Citado na página 22.

BURKOV, A. *The Hundred-Page Machine Learning Book*. Andriy Burkov, 2019. ISBN 9781999579517. Disponível em: <https://books.google.com.br/books?id=0jbxwQEACAAJ>. Citado 3 vezes nas páginas 16, 17 e 21.

CATON, S.; HAAS, C. Fairness in machine learning: A survey. *arXiv preprint arXiv:2010.04053*, 2020. Citado 2 vezes nas páginas 11 e 34.

CHEN, D. et al. Gan-leaks: A taxonomy of membership inference attacks against generative models. In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. [S.l.: s.n.], 2020. p. 343–362. Citado na página 35.

CHEN, H. et al. *Data-Free Learning of Student Networks*. 2019. Citado 4 vezes nas páginas 12, 13, 43 e 44.

CHEN, X. et al. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *arXiv preprint arXiv:1606.03657*, 2016. Citado 2 vezes nas páginas 33 e 34.

CHOWDHURY, M.; APON, A.; DEY, K. *Data analytics for intelligent transportation systems*. [S.l.]: Elsevier, 2017. Citado na página 16.

CLEVERT, D.-A.; UNTERTHINER, T.; HOCHREITER, S. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015. Citado na página 20.

COATES, A.; NG, A.; LEE, H. An analysis of single-layer networks in unsupervised feature learning. In: JMLR WORKSHOP AND CONFERENCE PROCEEDINGS. *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. [S.l.], 2011. p. 215–223. Citado 3 vezes nas páginas 37, 38 e 50.

COHEN, G. et al. Emnist: Extending mnist to handwritten letters. In: IEEE. *2017 International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2017. p. 2921–2926. Citado 3 vezes nas páginas 37, 38 e 50.

CORREIA-SILVA, J. R. et al. Copycat cnn: Stealing knowledge by persuading confession with random non-labeled data. In: IEEE. *2018 International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2018. p. 1–8. Citado 6 vezes nas páginas 12, 13, 14, 41, 42 e 49.

CORREIA-SILVA, J. R. et al. Copycat cnn: Are random non-labeled data enough to steal knowledge from black-box models? *Pattern Recognition*, Elsevier, v. 113, p. 107830, 2021. Citado 11 vezes nas páginas 12, 36, 41, 44, 46, 47, 48, 49, 52, 57 e 61.

DANG, N. C.; MORENO-GARCÍA, M. N.; PRIETA, F. De la. Sentiment analysis based on deep learning: A comparative study. *Electronics*, Multidisciplinary Digital Publishing Institute, v. 9, n. 3, p. 483, 2020. Citado na página 11.

DENG, J. et al. Imagenet: A large-scale hierarchical image database. In: IEEE. *2009 IEEE conference on computer vision and pattern recognition*. [S.l.], 2009. p. 248–255. Citado 12 vezes nas páginas 12, 26, 32, 33, 39, 43, 48, 50, 51, 54, 60 e 61.

DONAHUE, J.; KRÄHENBÜHL, P.; DARRELL, T. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016. Citado 2 vezes nas páginas 33 e 34.

DUMOULIN, V.; VISIN, F. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016. Citado na página 30.

ELMAN, J. L. Finding structure in time. *Cognitive science*, Wiley Online Library, v. 14, n. 2, p. 179–211, 1990. Citado na página 19.

GOODFELLOW, I. et al. *Deep learning*. [S.l.]: MIT press Cambridge, 2016. v. 1. Citado 9 vezes nas páginas 18, 19, 20, 21, 23, 24, 26, 28 e 47.

GOODFELLOW, I. J. et al. *Generative Adversarial Networks*. 2014. Citado 6 vezes nas páginas 11, 13, 29, 30, 33 e 49.

GU, J. et al. Recent advances in convolutional neural networks. *Pattern Recognition*, Elsevier, v. 77, p. 354–377, 2018. Citado na página 25.

HAGENDORFF, T. The ethics of ai ethics: An evaluation of guidelines. *Minds and Machines*, Springer, v. 30, n. 1, p. 99–120, 2020. Citado na página 11.

HAYES, J. et al. Logan: Membership inference attacks against generative models. *arXiv preprint arXiv:1705.07663*, 2017. Citado na página 35.

HAYKIN, S. S. et al. *Neural networks and learning machines/Simon Haykin*. [S.l.]: New York: Prentice Hall,, 2009. Citado na página 20.

HE, K. et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778. Citado 2 vezes nas páginas 27 e 47.

HEUSEL, M. et al. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *arXiv preprint arXiv:1706.08500*, 2017. Citado 2 vezes nas páginas 32 e 33.

HIBBARD, B. Ethical artificial intelligence. *arXiv preprint arXiv:1411.1373*, 2014. Citado na página 34.

HILPRECHT, B.; HÄRTERICH, M.; BERNAU, D. Monte carlo and reconstruction membership inference attacks against generative models. *Proceedings on Privacy Enhancing Technologies*, Sciendo, v. 2019, n. 4, p. 232–249, 2019. Citado na página 35.

HINTON, G. E.; OSINDERO, S.; TEH, Y.-W. A fast learning algorithm for deep belief

nets. *Neural computation*, MIT Press, v. 18, n. 7, p. 1527–1554, 2006. Citado na página 30.

HU, J.; SHEN, L.; SUN, G. Squeeze-and-excitation networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2018. p. 7132–7141. Citado 2 vezes nas páginas 27 e 48.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: PMLR. *International conference on machine learning*. [S.l.], 2015. p. 448–456. Citado na página 22.

ISLAM, S. R. et al. Explainable artificial intelligence approaches: A survey. *arXiv preprint arXiv:2101.09429*, 2021. Citado na página 34.

JABBAR, A.; LI, X.; OMAR, B. A survey on generative adversarial networks: Variants, applications, and training. *arXiv preprint arXiv:2006.05132*, 2020. Citado na página 29.

JAGIELSKI, M. et al. High accuracy and high fidelity extraction of neural networks. In: *29th {USENIX} Security Symposium ({USENIX} Security 20)*. [S.l.: s.n.], 2020. p. 1345–1362. Citado na página 36.

JIA, Y. et al. Caffe: Convolutional architecture for fast feature embedding. In: *Proceedings of the 22nd ACM international conference on Multimedia*. [S.l.: s.n.], 2014. p. 675–678. Citado na página 52.

JUUTI, M. et al. Prada: protecting against dnn model stealing attacks. In: IEEE. *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. [S.l.], 2019. p. 512–527. Citado na página 36.

KHAN, A. et al. A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, Springer, v. 53, n. 8, p. 5455–5516, 2020. Citado 3 vezes nas páginas 19, 22 e 27.

KRISHNA, K. et al. Thieves on sesame street! model extraction of bert-based apis. *arXiv preprint arXiv:1910.12366*, 2019. Citado na página 37.

KRIZHEVSKY, A.; HINTON, G. et al. Learning multiple layers of features from tiny images. Citeseer, 2009. Citado 6 vezes nas páginas 37, 38, 42, 43, 48 e 50.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, ACM New York, NY, USA, v. 60, n. 6, p. 84–90, 2017. Citado 2 vezes nas páginas 27 e 47.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, Springer Science and Business Media LLC, v. 521, n. 7553, p. 436–444, 05 2015. Disponível em: <https://doi.org/10.1038/nature14539>. Citado 2 vezes nas páginas 22 e 23.

LECUN, Y. et al. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, MIT Press - Journals, v. 1, n. 4, p. 541–551, dez. 1989. Disponível em: <https://doi.org/10.1162%2Fneco.1989.1.4.541>. Citado 2 vezes nas páginas 21 e 22.

LECUN, Y. et al. Handwritten digit recognition with a back-propagation network. In: _____. *Advances in Neural Information Processing Systems 2*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990. p. 396–404. ISBN 1558601007. Citado 2 vezes nas páginas 22 e 23.

LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, v. 86, n. 11, p. 2278–2324, Nov 1998. ISSN 0018-9219. Citado 2 vezes nas páginas 22 e 40.

LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, Ieee, v. 86, n. 11, p. 2278–2324, 1998. Citado 5 vezes nas páginas 37, 38, 40, 42 e 48.

LIU, S.; DENG, W. Very deep convolutional neural network based image classification using small training sample size. In: IEEE. *2015 3rd IAPR Asian conference on pattern recognition (ACPR)*. [S.l.], 2015. p. 730–734. Citado 2 vezes nas páginas 27 e 48.

LIU, Y. et al. Ml-doctor: Holistic risk assessment of inference attacks against machine learning models. *arXiv preprint arXiv:2102.02551*, 2021. Citado na página 36.

LUDERMIR, T. B.; SOUTO, M. C. de; OLIVEIRA, W. R. de. Weightless neural networks: knowledge-based inference system. In: IEEE. *2008 10th Brazilian Symposium on Neural Networks*. [S.l.], 2008. p. 207–212. Citado na página 19.

MAAS, A. L.; HANNUN, A. Y.; NG, A. Y. Rectifier nonlinearities improve neural network acoustic models. In: CITESEER. *Proc. icml*. [S.l.], 2013. v. 30, n. 1, p. 3. Citado na página 20.

MAO, Q. et al. Mode seeking generative adversarial networks for diverse image synthesis. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2019. p. 1429–1437. Citado 4 vezes nas páginas 31, 32, 51 e 52.

MASCI, J. et al. Stacked convolutional auto-encoders for hierarchical feature extraction. In: SPRINGER. *International conference on artificial neural networks*. [S.l.], 2011. p. 52–59. Citado na página 30.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943. Citado na página 18.

MILLI, S. et al. Model reconstruction from model explanations. In: *Proceedings of the Conference on Fairness, Accountability, and Transparency*. [S.l.: s.n.], 2019. p. 1–9. Citado na página 36.

MINSKY, M.; PAPERT, S. A. *Perceptrons: An introduction to computational geometry*. [S.l.]: MIT press, 2017. Citado na página 19.

MIRZA, M.; OSINDERO, S. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. Citado 4 vezes nas páginas 33, 34, 49 e 51.

MOSAFI, I.; DAVID, E. O.; NETANYAHU, N. S. Deepmimic: Mentor-student unlabeled data based training. In: SPRINGER. *International Conference on Artificial Neural Networks*. [S.l.], 2019. p. 440–455. Citado na página 42.

MOSAFI, I.; DAVID, E. O.; NETANYAHU, N. S. Stealing knowledge from protected deep neural networks using composite unlabeled data. In: IEEE. *2019 International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2019. p. 1–8. Citado 2 vezes nas páginas 42 e 44.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: *Icml*. [S.l.: s.n.], 2010. Citado na página 20.

NETZER, Y. et al. Reading digits in natural images with unsupervised feature learning. 2011. Citado 4 vezes nas páginas 37, 38, 48 e 50.

NIELSEN, M. A. *Neural networks and deep learning*. [S.l.]: Determination press San Francisco, CA, 2015. v. 25. Citado 2 vezes nas páginas 18 e 19.

ODENA, A. Semi-supervised learning with generative adversarial networks. *arXiv preprint arXiv:1606.01583*, 2016. Citado 2 vezes nas páginas 33 e 34.

ODENA, A.; OLAH, C.; SHLENS, J. Conditional image synthesis with auxiliary classifier gans. In: PMLR. *International conference on machine learning*. [S.l.], 2017. p. 2642–2651. Citado 3 vezes nas páginas 33, 34 e 49.

OH, S. J.; SCHIELE, B.; FRITZ, M. Towards reverse-engineering black-box neural networks. In: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. [S.l.]: Springer, 2019. p. 121–144. Citado 2 vezes nas páginas 36 e 37.

OREKONDY, T.; SCHIELE, B.; FRITZ, M. Knockoff nets: Stealing functionality of black-box models. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2019. p. 4954–4963. Citado 3 vezes nas páginas 41, 42 e 52.

PAPERNOT, N.; MCDANIEL, P.; GOODFELLOW, I. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016. Citado na página 40.

PAPERNOT, N. et al. *Practical Black-Box Attacks against Machine Learning*. 2017. Citado 2 vezes nas páginas 38 e 40.

PASZKE, A. et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. Citado na página 52.

PRATI, R. C. *Novas abordagens em aprendizado de máquina para a geração de regras, classes desbalanceadas e ordenação de casos*. Tese (Doutorado) — Universidade de São Paulo, 2006. Citado 2 vezes nas páginas 16 e 17.

RAMSUNDAR, B.; ZADEH, R. B. *TensorFlow for deep learning: from linear regression to reinforcement learning*. [S.l.]: "O'Reilly Media, Inc.", 2018. Citado 4 vezes nas páginas 23, 24, 25 e 26.

RASHED, E.; SEOUD, M. S. A. E. Deep learning approach for breast cancer diagnosis. In: *Proceedings of the 2019 8th International Conference on Software and Information Engineering*. [S.l.: s.n.], 2019. p. 243–247. Citado na página 11.

RAWAT, W.; WANG, Z. Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation*, v. 29, p. 1–98, 06 2017. Citado na página 23.

RAZAVIAN, A. S. et al. Cnn features off-the-shelf: an astounding baseline for recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. [S.l.: s.n.], 2014. p. 806–813. Citado na página 28.

RIGAKI, M.; GARCIA, S. A survey of privacy attacks in machine learning. *arXiv preprint arXiv:2007.07646*, 2020. Citado 6 vezes nas páginas 12, 34, 35, 36, 37 e 47.

ROSENBLATT, F. *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. [S.l.], 1961. Citado na página 18.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. *Learning internal representations by error propagation*. [S.l.], 1985. Citado na página 21.

RUSSAKOVSKY, O. et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, Springer, v. 115, n. 3, p. 211–252, 2015. Citado 2 vezes nas páginas 26 e 39.

RUSSELL, S.; NORVIG, P. Artificial intelligence: a modern approach. 2002. Citado 3 vezes nas páginas 16, 17 e 18.

SALIMANS, T. et al. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016. Citado na página 32.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, IBM, v. 3, n. 3, p. 210–229, 1959. Citado 2 vezes nas páginas 11 e 16.

SHI, Y.; SAGDUYU, Y.; GRUSHIN, A. How to steal a machine learning classifier with deep learning. In: IEEE. *2017 IEEE International Symposium on Technologies for Homeland Security (HST)*. [S.l.], 2017. p. 1–5. Citado 4 vezes nas páginas 12, 13, 40 e 44.

SHOKRI, R. et al. Membership inference attacks against machine learning models. In: IEEE. *2017 IEEE Symposium on Security and Privacy (SP)*. [S.l.], 2017. p. 3–18. Citado na página 35.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. Citado 2 vezes nas páginas 27 e 47.

SRIVASTAVA, N. et al. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, JMLR. org, v. 15, n. 1, p. 1929–1958, 2014. Citado na página 19.

SZANDAŁA, T. Review and comparison of commonly used activation functions for deep neural networks. In: *Bio-inspired Neurocomputing*. [S.l.]: Springer, 2020. p. 203–224. Citado na página 19.

SZE, V. et al. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, Ieee, v. 105, n. 12, p. 2295–2329, 2017. Citado 3 vezes nas páginas 20, 21 e 22.

SZEGEDY, C. et al. Going deeper with convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2015. p. 1–9. Citado na página 27.

SZEGEDY, C. et al. Rethinking the inception architecture for computer vision. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 2818–2826. Citado na página 32.

TRAMÈR, F. et al. Stealing machine learning models via prediction apis. In: *25th {USENIX} Security Symposium ({USENIX} Security 16)*. [S.l.: s.n.], 2016. p. 601–618. Citado 5 vezes nas páginas 12, 13, 36, 40 e 44.

UTGOFF, P. E.; STRACUZZI, D. J. Many-layered learning. *Neural computation*, MIT Press, v. 14, n. 10, p. 2497–2529, 2002. Citado na página 21.

WANG, X. et al. The security of machine learning in an adversarial setting: A survey. *Journal of Parallel and Distributed Computing*, Elsevier, v. 130, p. 12–23, 2019. Citado na página 34.

XU, Q. et al. An empirical study on evaluation metrics of generative adversarial networks. *arXiv preprint arXiv:1806.07755*, 2018. Citado na página 32.

YANG, X.; WANG, X. Recognizing license plates in real-time. *arXiv preprint arXiv:1906.04376*, 2019. Citado na página 11.

YOSINSKI, J. et al. How transferable are features in deep neural networks? *arXiv preprint arXiv:1411.1792*, 2014. Citado 2 vezes nas páginas 27 e 28.

YUAN, X. et al. *ES Attack: Model Stealing against Deep Neural Networks without Data Hurdles*. 2020. Citado 3 vezes nas páginas 12, 33 e 43.

ZEILER, M. D.; FERGUS, R. Visualizing and understanding convolutional networks. In: SPRINGER. *European conference on computer vision*. [S.l.], 2014. p. 818–833. Citado na página 27.

ZEILER, M. D. et al. Deconvolutional networks. In: IEEE. *2010 IEEE Computer Society Conference on computer vision and pattern recognition*. [S.l.], 2010. p. 2528–2535. Citado na página 30.

ZENG, X. et al. Crafting gbd-net for object detection. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 40, n. 9, p. 2109–2123, 2017. Citado na página 27.

ZHANG, R. et al. The unreasonable effectiveness of deep features as a perceptual metric. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2018. p. 586–595. Citado na página 32.