Universidade Estadual de Londrina

MELVI TREVISAN FERREIRA

DETECÇÃO DE ANOMALIAS EM REDE UTILIZANDO *AUTOENCODERS* E REDES ADVERSÁRIAS GENERATIVAS

LONDRINA
2022

MELVI TREVISAN FERREIRA

DETECÇÃO DE ANOMALIAS EM REDE UTILIZANDO *AUTOENCODERS* E REDES ADVERSÁRIAS GENERATIVAS

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Mario Lemes Proença Jr.

LONDRINA

2022

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UEL

Ferreira, Melvi Trevisan.
Detecção de anomalias em rede utilizando Autoencoders e Redes Adversárias Generativas / Melvi Trevisan Ferreira. - Londrina, 2022.
62 f.

Orientador: Prof. Dr. Mário Lemes Proença.
Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Universidade Estadual de Londrina, Centro de Ciências Exatas, Graduação em Ciência da Computação, 2022.
Inclui bibliografia.

1. Inteligência artificial - TCC. 2. Redes Autoencoder - TCC. 3. Redes GAN - TCC. 4. Anomalias em rede de computadores - TCC. I. Proença, Prof. Dr. Mário Lemes. II. Universidade Estadual de Londrina. Centro de Ciências Exatas. Graduação em Ciência da Computação. III. Título.

CDU 519

MELVI TREVISAN FERREIRA

DETECÇÃO DE ANOMALIAS EM REDE UTILIZANDO *AUTOENCODERS* E REDES ADVERSÁRIAS GENERATIVAS

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA

Orientador: Prof. Dr. Mario Lemes Proença Jr.
Universidade Estadual de Londrina

Prof. Dr. Fábio Sakuray
Universidade Estadual de Londrina

Ms. Daniel Matheus Brandão Lent
Universidade Estadual de Londrina

Londrina, 31 de maio de 2022.

AGRADECIMENTOS

Este é o encerramento de uma jornada, o encerramento de um longo capítulo de 4.. *(COVID-19)*.. 5 anos na Universidade Estadual de Londrina. Definitivamente, é muita coisa para se olhar pra trás e pensar. Tijolos que dia a dia foram colocados aos poucos pelas mais diversas pessoas até chegar hoje, até chegar este momento em que escrevo estes agradecimentos e envio esse trabalho para a coordenação.

Gostaria de agradecer inicialmente aos meus colegas de curso, que acompanharam mais de perto que qualquer outra pessoa tudo até aqui. Especialmente, ao Vinicius, Matheus e Felipe, nestes últimos anos mais conturbados de curso.

Agradeço também aos meus pais, que sempre proveram tudo ao seu alcance para que eu pudesse chegar aqui, sempre colocando a educação como algo indispensável. Em especial, à Zely, que com certeza, moldou muito do que eu sou hoje.

Agradeço ao professor Mario, meu orientador, não só pelo convite mas por toda compreensão e atenção que foi dedicada durante o desenvolvimento deste trabalho.

Por fim, também agradeço ao professor Luiz Fernando Carvalho pela diferença que ele fez no meu primeiro ano do curso e tudo que se seguiu desde então.

"Não tenho medo de errar,
somente de desistir."

FERREIRA, M. T.. **Detecção de anomalias em rede utilizando *Autoencoders* e Redes Adversárias Generativas**. 2022. 62f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2022.

RESUMO

Hoje, a Internet se consolidou como uma necessidade indispensável. Sua constante evolução para atender as demandas das novas tecnologias e paradigmas que, agora dependem dela, findaram um cenário que não deve mudar. A estabilidade e segurança dessa rede, cada vez mais complexa, passa a exigir um controle humanamente impossível. É frente a esse panorama que este trabalho se propõe a projetar um sistema de detecção de intrusão e anomalias. Em específico, explorando duas ramificações promissoras das Redes Neurais de Aprendizagem Profunda: *Autoencoders* e Redes Adversárias Generativas. Como resultado, foram criados 3 modelos de redes neurais profundas diferentes e avaliadas suas métricas com dois conjuntos de dados com ataques *DDoS* (ataques de negação de serviço) e *PortScan* (ataques de escaneamento de portas). Por fim, os resultados demonstraram uma alto desempenho nas métricas utilizadas, especialmente pela rede GAN em relação as outras. Dessa forma, validando o potencial sugerido das arquiteturas testadas para a detecção de anomalias com sucesso.

Palavras-chave: Detecção de anomalias. Autoencoder. GAN. Deep learning. Aprendizado de máquina. Redes neurais profundas

FERREIRA, M. T.. **Anomaly detection using Autoencoders and Generative Adversarial Networks**. 2022. 62p. Final Project (Bachelor of Science in Computer Science) – State University of Londrina, Londrina, 2022.

ABSTRACT

Nowadays, the internet has become an indispensable necessity. The evolution of the network to meet demands from the new technologies and paradigms that now depend on it shaped a future that doesn't exist without it. Keeping the stability and security of this complex, growing network isn't humanly possible anymore. That's the scenario where this work proposes to project a system to detect intrusions and anomalies. Specifically, exploring two promising fields of Deep Learning: Generative adversarial networks and Autoencoders. As a result, 3 different deep neural network models were created and their metrics were evaluated with two datasets with *DDoS* (distributed denial of service) and *PortScan* attacks. Finally, the results showed a high performance in the metrics used, especially by the GAN network in relation to the others. Thus, validating the suggested potential of the tested architectures for successful anomaly detection.

Keywords: Anomaly detection. Autoencoder. GAN. Deep learning. Machine Learning. Deep neural networks

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACODS	Ant Colony Optimization for Digital Signature
AIS-IDS	*Artificial Immune System based IDS*
ANN	*Artificial neural networks*
CkNN	*Continuous K-Nearest Neighbor*
CNN	*Convolutional Neural Network*
DDoS	*Distributed Denial of Service*
DL	*Deep Learning*
DNN	*Deep neural networks*
DoS	*Denial of Service*
DSNSF	*Digital Signature of Network Segment using Flow*
EGBAD	*Efficient GAN-Based Anomaly Detection*
FSN	*Finite-state Machine*
GAN	*Generative Adversarial Networks*
GRU	*Gated Recurrent Units*
HIDS	*Network-based Intrusion Detection System*
ICMP	*Internet Control Message Protocol*
IDS	*Intrusion Detection System*
IoT	*Internet of Things*
K-NN	*K-nearest Neighbors*
LSTM	*Long Short-term Memory*
MLP	*Multi Layer Perceptron*
NIDS	*Host-based Intrusion Detection System*
NSA	*Negative Selection Algorithm*
PCA	*Principal Component Analysis*

PSO	*Particle Swarm Optimization*
SDN	*Software Defined Network*
SSH	*Secure Socket Shell*
SVM	*Support Vector Machine*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
VAE	*Variational Autoencoder*

LISTA DE SÍMBOLOS

Σ	Letra grega Sigma, maiscúla, indica somatório
$H()$	Função de Entropia
$max()$	Função de máximo de um conjunto
$min()$	Função de mínimo de um conjunto

SUMÁRIO

1 INTRODUÇÃO

Nos dias atuais, a internet incontestavelmente tornou-se uma necessidade indispensável [1]. Um cenário que vem mudando drasticamente não só a dinâmica do nosso mundo, como também, da própria rede [2]. Novos paradigmas e tecnologias como a Internet das Coisas (IoT), cidades inteligentes e veículos autônomos vêm surgindo e se estabelecendo mais a cada dia em direção ao futuro, e todos eles possuem algo em comum: se apoiam na existência dessa rede global de comunicação [3].

As redes de computadores mudaram a forma como as pessoas realizam suas tarefas diárias. Operações como movimentações de contas bancárias, comunicações por voz, transmissões de vídeo, acesso a notícias, entre muitas outras atividades, passaram a ser feitas via internet [4]. Consequentemente, essa crescente demanda se refletiu no volume e complexidade do tráfego. Logo, a manutenção das redes de larga escala se tornaram um desafio no qual as técnicas e métodos tradicionais passaram a ser inviáveis [5, 2, 6].

Em resposta a isto, a pesquisa ao longo dos anos levou ao estado da arte que, hoje, pode ser observado em duas vanguardas que emergiram como paradigmas para esta nova geração: Redes definidas por *Software* (*SDN*) [7, 6, 8] e Aplicação de aprendizado profundo de máquina (*Deep Learning*) para detecção de anomalias e intrusões a redes [9, 3, 10].

A segurança das redes se tornou uma necessidade real. A grande quantidade de informações que agora transitam nelas deu origem a inúmeras ameaças, de simples vírus a complexas invasões e roubos de dados [11]. Assim, o monitoramento do tráfego e a detecção de anomalias passaram a ser indispensáveis [12]. Neste campo, a pesquisa passou a se voltar para o uso da inteligência artificial, mais recentemente, com o uso do aprendizado profundo (*Deep Learning*). Seu sucesso e performance superior quando comparados ao *Shallow Learning* em outras áreas têm igualmente se demonstrado verdadeiros quando aplicados na detecção de anomalias [1, 13, 9].

É importante lembrar que, embora a área de detecção de anomalias já venha sendo explorada com os mais diversos métodos nos últimos anos ao longo de inúmeros trabalhos (*Adversarial Deep Learning* [14], *GRU* [15], *CNN* [16], *Artificial Immune Systems* [2], *LSTM* [3], *Discrete Wavelet Transform* [17], *Genetic Algorithm* [4], *Paraconsistent Machine* [18], *Multinomial Logistic Regression* [7], *ACODS* [12], *Holt-Winters* [8], entre outros), ela continua sendo um campo aberto de pesquisa. Apesar dos inúmeros esforços, as soluções atuais não têm sido suficientes para atender as demandas devido a grande quantidade de diferentes cenários, redes e arquiteturas com que se pode lidar [15, 13]. Outro fator que também contribui para isso é a crescente complexidade do tráfego [12],

juntamente das constantes mudanças da rede e sua variabilidade [19].

No aspecto operacional, a arquitetura tradicional de administração das redes se tornou inadequada e incapaz de acompanhar seu crescimento, fato em especial agravado pelo advento da computação em nuvem [3]. As políticas de rede que continuaram as mesmas por décadas simplesmente não eram sofisticadas o suficiente para trabalhar com as diferentes condições e problemas que as massivas quantidade de dados trouxeram [9].

Neste contexto, as Redes Definidas por Software (*SDN*) surgiram com uma alternativa promissora, permitindo uma escalabilidade e flexibilidade sem precedentes [8, 2]. Nesta arquitetura, a separação em camadas entre os dados que trafegam nela dos mecanismos que executam o controle operacional, juntamente com a sua centralização operacional, permitiram que as redes fossem controladas programaticamente com um âmbito mais universal e intuitivo em contraposição da até então configuração específica de cada aparelho [6, 20].

Neste trabalho, será estudado a aplicação de duas ramificações promissoras do aprendizado profundo de máquina na detecção de anomalias: Redes Adversárias Generativas e *Autoencoders*. O objetivo é o desenvolvimento de redes neurais que explorem as arquiteturas em questão, assim como, a avaliação de seus desempenhos. Dessa forma, este documento compreende no capítulo 2, definições de conceitos básicos como anomalias, seus escopos e teorias além de intrusões de rede. No capítulo 3, abrange uma breve introdução a inteligência artificial, suas bases, assim como as teorias das duas redes principais aqui aplicadas. No capítulo 4, traz definições formais de sistemas de detecção de intrusões, características de ataques e trabalhos relacionados. No capítulo 5, apresenta os trabalhos relacionados. No capítulo 6, apresenta os modelos de redes neurais desenvolvidos assim como toda metodologia de avaliação e cenários de teste. No capítulo 7, por final, discute as conclusões a partir dos resultados esperados e obtidos.

2 ANOMALIAS

Destrinchando o título deste trabalho, "Detecção de anomalias em redes *[utilizando Autoencoders e Redes Adversárias Generativas]*", se torna evidente o papel central que as anomalias possuem nele e, assim sendo, parece prudente tomar como um ponto de partida definir exatamente o que elas são e o que estamos buscando detectar.

Formalmente, o termo anomalia traz várias definições, sendo uma área de pesquisa desde o começo do século XIX. Neste tempo, os inúmeros estudos e pesquisas feitas trouxeram com eles diferentes visões e especificidades ao tema, especialmente, de acordo com o domínio da aplicação [21, 22].

Em uma visão ampla pela literatura, Barnett e Lewis definem anomalia como "uma observação (ou conjunto de observações) que parece ser inconsistente com o resto de seu conjunto de dados" [23]. Hawkins classifica-a como "uma observação que se desvia o suficientemente de outras para que se seja levantado suspeitas de que ela foi gerada por um mecanismo diferente das demais" [24]. Chandola et al., de forma mais categórica, afirma que "anomalias são padrões de dados que não se comportam como o esperado" [25]. Hoque et al. as descreve como "padrões que não se conformam quando comparados a uma noção bem definida de normalidade" [26].

Dessa forma, embora existam mais definições, já torna-se claro uma convergência entre suas caracterizações: um dos principais passos para a caracterização de uma anomalia é justamente definir o seu oposto, em outra palavras, definir a normalidade [27, 22].

2.1 Anomalias de rede

Neste contexto, este trabalho se focará exclusivamente nas anomalias observadas nas redes de computadores. Seguindo as definições apresentadas, podemos afirmar que as anomalias englobam tudo que não se caracteriza como "normal" na rede. Podendo elas serem classificadas baseado em várias características diferentes de acordo com o interesse do tópico e da pesquisa [21, 28].

2.1.1 Categorizações e tipos

De acordo com Fernandes et al. [27], anomalias de rede de forma simplificada podem ser categorizadas baseadas em duas propriedades relevantes: sua natureza e seu aspecto causal, independentemente de serem maliciosas ou não, conforme ilustrado na Tabela 1.

Partindo de sua categorização mais básica, a intenção de uma anomalia diz respeito

Tabela 1 – Categorização de anomalias de tráfego

Intenção	**Natureza**	**Aspecto Causal**
Maliciosa	Pontual	Operacional
Não maliciosa	Coletiva	Flash Crowd
	Contextual	Medição
		Ataque de rede

se determinada ocorrência representa uma ameaça e/ou tem intenção prejudicial ou não. Essa classificação embora seja geralmente implícita como maliciosa, faz-se necessária para delimitar o escopo de um detecção, dado que as anomalias também incluem eventos não esperados ou ruídos presentes na rede que não são lesivos.

Sua segunda categorização possível é de acordo com sua natureza. Esta propriedade diz respeito com quando uma determinada observação ou dado passa a ser considerada uma anomalia e possui três categorias:

1. Uma anomalia é considerada pontual quando uma instância individual de dados se desvia do padrão, sendo ela o caso mais simples possível e o maior foco da maioria das pesquisas da área [25]. Nesta situação, a própria ocorrência já se destoaria sem maiores dificuldades, como por exemplo um pico de tráfego na rede.

2. Uma anomalia coletiva ocorre quando uma determinada coleção de dados se comporta diferentemente do resto de seu conjunto. Neste caso, as suas ocorrências e dados individuais não necessariamente se caracterizam uma anomalia, mas se apresentam como uma quando observados em conjunto [22]. Analogamente, uma tentativa de acesso mal sucedida vindo de um país exterior não necessariamente representaria uma anomalia, mas um conjunto de várias claramente se configuraria como um ataque.

3. Uma anomalia contextual ou condicional depende do contexto em que ela foi encontrada para ser caracterizada como tal. Para isso, as condições ou contexto devem ser previamente formuladas e enumeradas [29]. Um exemplo poderia ser uma conexão *SSH* realizada fora horário de expediente do trabalho, sendo dependente da condicional de determinado horário para se caracterizar uma anomalia.

Por último, uma terceira classificação possível se trata do aspecto causal de anomalia. É uma categoria que trata de sua origem, podendo ser ela operacional, *flash crowd* (multidão relâmpago), de medição ou ataque [30, 28]. Respectivamente, essas anomalias podem ser representadas por:

- erros de configuração ou falhas de rede, incluindo problemas de hardware e software em qualquer lugar de sua cadeia de operação [31];

- picos de tráfego de acessos legítimos, como em grandes promoções e vendas ou anúncios esperados;
- problemas na coleta ou medição dos dados e estatísticas da rede e
- ataques intencionalmente maliciosos de qualquer tipo, como *worms*, abusos de funcionalidade, *DDoS*, entre outros que visem comprometer a integridade, confidencialidade ou disponibilidade da rede [32].

2.2 Intrusões de rede

Uma vez feita essa categorização, se torna possível observar a necessidade de definir o ponto de interesse na criação de um sistema que possa detectar tais anormalidades. Observe que embora, como mostrado, as anomalias compreendam muito mais que eventos maliciosos, estes outros casos usualmente não se enquadram no escopo de interesse de sistemas de proteção de rede, assim, não devendo ser reportados, dado que ainda que sejam anomalias, se tratam de tráfego legítimo.

Por conseguinte, o resultado é que o foco avance e se reposicione para não detectar meramente anomalias, mas sim, anomalias que representem intrusões de rede.

2.3 Ataques de rede

Um ataque de rede pode ser definido como qualquer ação maliciosa que vise interromper, negar, degradar ou destruir informações e serviços em uma rede de computadores [32, 22] através do fluxo de dados da rede com objetivo de comprometer a integridade, confidencialidade ou disponibilidade dela.

Sendo uma definição extremamente abrangente, Hansman [33], em sua taxonomia conforme na Tabela 2, os divide entre primariamente entre 9 subclasses: *Vírus*, *Worms*, *Trojans*, *Buffer overflows*, *DoS*, Ataques de rede, Ataques físicos, Ataques a senha e Ataques de obtenção de informação. Sendo esta uma das mais tipicamente utilizadas na área.

Dentre estes, alguns exemplos de ataques extremamente comuns e estudados são os *DDoS* (ataque distribuído de negação de serviço) e *PortScan* (escaneamento de portas), respectivamente, representando os ataques de negação de serviço (*DoS*) e de obtenção de informações na primeira dimensão de Hansman [27].

2.3.1 *DDoS* - Negação distribuída de serviço

Os ataques DoS (*Denial of Service*) possuem como principal característica a degradação ou interrupção total do acesso de usuários legítimos a uma rede ou serviço [34].

Tabela 2 – Primeira dimensão da categorização de ataques de Hansman

Classificação	
Virus	Programas auto replicantes que se propagam através da contaminação de arquivos ou programas já existentes
Worm	Programas auto replicantes que se propagam utilizando-se de serviços na rede
Trojan	Programas maliciosos que se escondem dentro de programas aparentemente benignos
Buffer overflow	Processos que ganham controle ou derrubam outros processos sobrecarregando os limites de um *buffer*
DoS	Ataques que impedem usuários legítimos de acessar uma rede ou servidor
Ataque de rede	Ataques focados em uma rede ou seus usuários através da manipulação de protocolos em qualquer uma de suas camadas
Ataque físico	Ataques que afetam os componentes físicos de uma rede ou computador
Ataque a senha	Ataques que buscam obter senhas, geralmente, através da tentativa bruta de vários acessos
Ataque de obtenção de informação	Ataques que buscam obter informações ou vulnerabilidades através de um escaneamento; apesar de não causarem danos de imediato, concedem informações importantes para ataques futuros

Este tipo de ataque opera com o esgotamento dos recursos computacionais do alvo, sobrecarregando-o com requisições acima de sua capacidade de processamento. Usualmente, ocorrendo através de grande volume de tráfego irrelevante sendo enviado à vítima para congestionar e consumir processamento que deveria estar atendendo a requisições legítimas.

Quando estes ataques ocorrem de forma distribuída, utiliza-se o termo DDoS (*Distributed Denial of Service*). Nesta forma, ele passa a ser executado a partir de diversas origens ao mesmo tempo com o objetivo tanto de obter um fluxo maior de dados para o ataque quanto dificultar medidas protetivas contra o mesmo [35].

Exemplos comuns de ataques *DDoS* ocorrem através de *botnets*, casos cada vez mais impulsionados com o crescimento do *IoT* devido a alta negligência da segurança dos dispositivos conectados a rede [36]. Nesta execução, cria-se um rede massiva de dispositivos infectados que aguardam silenciosamente comandos para, sincronizadamente em

determinado momento, disparar um ataque contra um determinado alvo.

Hoje, os ataques *DDoS* tem se tornado uma das principais ameaças, apresentando números cada vez mais expressivos ano após ano. Relatórios mostram que empresas chegam a perder mais de $100.000 por hora [37] devido a ataques como este. Demonstrando como claramente se trata de um dos maiores desafios de segurança da atualidade.

2.3.2 *PortScan* - Escaneamento de portas

Os ataques *PortScan (Escaneamento de Portas)* possuem como principal características a obtenção de informações sobre um alvo. Como o próprio nome sugere, é feito um escaneamento de quais portas estão abertas ou respondem solicitações em um determinado alvo enviando pacotes para todas elas [38].

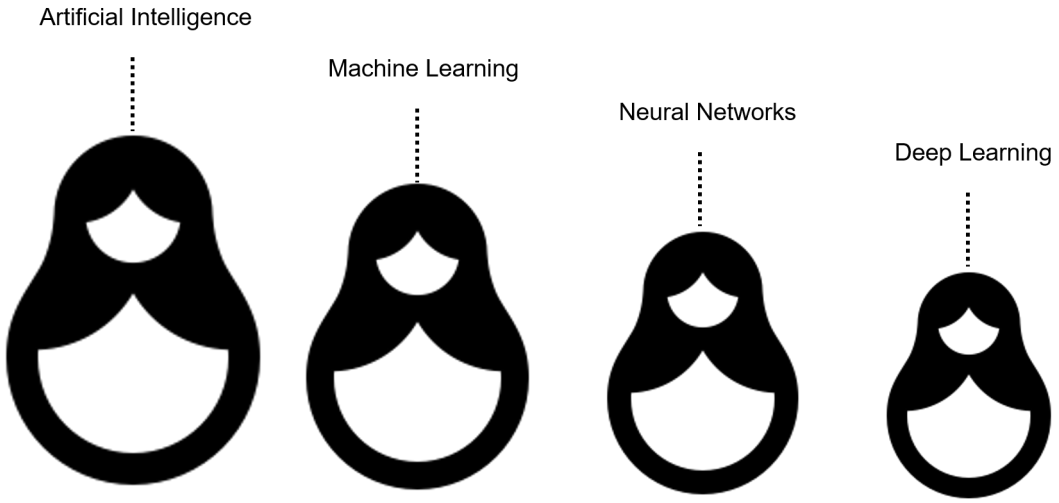
Apesar de em primeiro momento não causar danos, a execução desse ataque proporciona uma vasta quantidade de informações sobre o alvo para futuros ataques. É possível detectar detalhes como qual o sistema operacional sendo executado, quais as aplicações que estão sendo executadas nele, entre outros, assim, proporcionando uma lista de possíveis vetores de ataque [39].

Isso geralmente se torna possível pelo seguimento de portas padrões determinadas aplicações: como servidores do banco de dados MySQL utilizam a porta 3306 por padrão, um *PortScan* que retorne que esta porta está aberta muito provavelmente confirma a existência de um banco de dados aberto para um possível ataque a ser executado.

3 INTELIGÊNCIA ARTIFICIAL, MACHINE LEARNING E REDES NEURAIS

A inteligência artificial se tornou um tópico ubíquo. Impulsionada pelas grandes quantidades de dados disponíveis para serem analisados, ela não só se tornou uma das maiores apostas para o futuro da computação como também começou a redefinir padrões e operações nas mais diversas áreas [40, 41, 42, 43].

McCarthy [44] define a inteligência artificial como "a ciência e engenharia de fazer máquinas e, especialmente, programas de computadores inteligentes". Hoje, ela é principalmente representada pelo ramo do *Machine Learning*. Na qual, conforme mostrado na Figura 1, ainda se expande para vários campos: um deles, sendo as Redes Neurais e, mais adiante ainda, se espalhando em mais direções até chegar ao *Deep Learning*.



Fonte: [45]

Figura 1 – Escopo de termos comuns a inteligência artificial

3.1 Machine learning

Aprendizado de máquina ou *Machine Learning* pode ser definido como um método computacional que usa a experiência (dados previamente coletados) para melhorar a performance de algo ou fazer predições precisas. Se trata de um sub-campo da inteligência artificial onde se busca criar algoritmos que possam automaticamente se modificar e adaptar sua arquitetura através da repetição (treinamento) e experiência (resultados prévios de sua própria execução ou dados rotulados), assim, evoluindo e obtendo melhores resultados na tarefa desejada [46, 47, 48, 49].

De forma geral, as técnicas de aprendizado de máquina aplicadas na detecção de anomalias hoje em dia podem ser divididas em duas frentes: *Shallow Learning* e *Deep*

Learning [50, 3]. Na tabela 3, é possível observar alguns exemplos mais comuns de cada categoria.

Shallow learning é o termo utilizado para modelos mais tradicionais do aprendizado de máquina, englobando algoritmos como *Random Forest*, SVM (*Support Vector Machine*), K-NN (*k-nearest neighbors*), entre outros além das próprias redes neurais com poucas camadas ocultas.

Tabela 3 – Exemplos de algoritmos de aprendizado de máquina

Shallow Learning	**Deep Learning**
Random Forest	*Convolutional Neural Network*
PCA	*Deep Auto Encoder*
K-NN	*Long Short-Term Memory*
Algoritmo genético	*Generative Adversarial Network*
SVM	*Recurrent Neural Network*

O objetivo da aplicação do aprendizado de máquina na detecção de anomalias é que se possa utilizá-lo para aprender padrões a partir de um amostra de dados e então, fazer predições ou classificações baseados nisso. Nesta área, dentro do Aprendizado de Máquina, é que entram as redes neurais. Especificamente, as redes neurais artificiais (*ANN*) que tentam imitar o funcionamento do cérebro humano [45].

3.2 Redes neurais artificiais

Uma rede neural artificial (*ANN*) é descrita por Ghorbani [32] como um modelo computacional com 4 elementos primordiais: um conjunto de unidades de processamento (neurons), um conjunto de sinapses (conexões e pesos), um padrão de conexão entre eles e um processo de aprendizado para treinar esta rede. Em outras palavras, se traduzindo em uma coleção de neurônios que são altamente interconectados dentro de uma determinada topologia. Nela, eles possuem a habilidade de aprender baseado em exemplos e de criar generalizações baseados em dados incompletos, ruidosos e limitados [51].

Na prática, isso pode ser observado de forma simples na Figura 2. Uma rede é primordialmente composta por uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Cada unidade de processamento (neuron), é representada por uma esfera azul e suas sinapses são representadas pelas conexões existentes entre elas.

Durante seu processamento, os dados são fornecidos na camada de entrada e percorrem a rede até sua camada de saída. Cada neurônio, por sua vez, recebe os dados através de suas sinapses e tem um peso atribuído a cada uma delas. Assim, ele é capaz de aplicar condições diferentes para cada conexão e efetuar um determinado processamento e lógica próprias para definir seu resultado. Então, repassando seu valor para as próximas

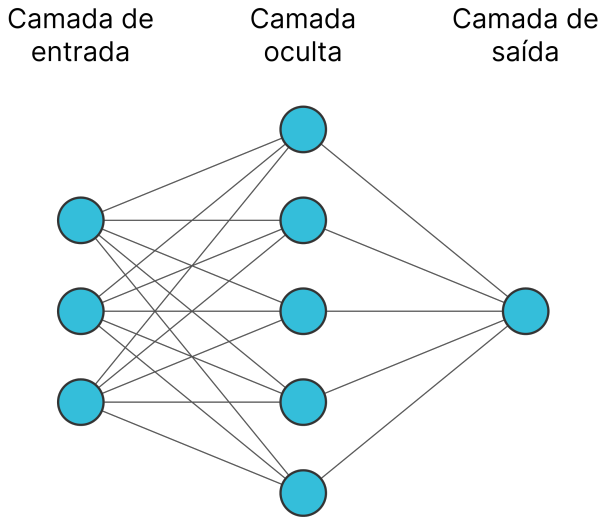


Figura 2 – Exemplo de rede neural artificial

camadas a frente que, continuamente, repetirão o mesmo processo até chegar na camada final de saída.

Durante o treinamento, o objetivo é que com base nas métricas utilizadas para comparar entre o resultado desejado e o resultado obtido, a rede lentamente atualize seus valores e teste suas mudanças para verificar se ela está mais perto de seu objetivo. Em redes mais simples, esses valores em questão correspondem aos os pesos atribuídos nas ligações entre os neurônios.

3.3 Redes neurais profundas

As Redes Neurais de Aprendizado Profundo, também referidas como *Deep Learning*, se tratam de um tipo específico de *Machine Learning*, dentro das redes neurais, que vem ganhando extrema notoriedade devido ao seu poder e flexibilidade em comparação com as redes tradicionais (*Shallow Learning*) [52, 3]. Especialmente, graças a sua capacidade de lidar com representações significativas de dados complexos e multidimensionais. Pode-se afirmar que essas redes profundas são bio-inspiradas no sentido de que, assim como o cérebro humano, elas arquiteturalmente se conectam de forma profunda entre seus neurônios [53].

Em comparação, os algoritmos *Shallow Learning* possuem limitações, como a sua considerável dependência dos atributos utilizados no treinamento, exigindo uma análise minuciosa para que se possa utilizar as melhores estatísticas da rede em seu treinamento [54]. Por sua vez, os algoritmos de *Deep Learning* conseguem realizar esta escolha de atributos (*features*) automaticamente, através do uso de várias camadas ocultas não lineares [1, 55]. Dessa forma, possuindo uma capacidade de aprendizado e generalização extremamente superior [56, 57].

Estruturalmente, os modelos de aprendizado profundo são usualmente construídos como redes de múltiplas camadas (*Multilayer Perceptron*). Essas múltiplas camadas geralmente são conhecidas como camadas ocultas, nas quais, diferentemente do aprendizado normal, ocorre a extração automática das características desejadas [1]. Por causa disso, muitas vezes são referidos como caixas pretas, onde embora tenham o desempenho conhecido, não se sabe exatamente como funcionam [55].

3.3.1 Redes GAN

Apresentada em 2014 por Goodfellow [58], as Redes Adversárias Generativas (*Generative Adversarial Nets*) se tratam de uma arquitetura de rede neural onde duas redes são postas de forma adversária uma a outra. A proposta é que uma rede generativa seja colocada contra uma rede discriminativa: respectivamente, enquanto uma tentará criar dados que imitem valores reais, a outra tentará classificar se determinada amostra é real ou foi gerada por sua rede adversária [14, 59]. Essa é uma abordagem que busca, dado um determinado conjunto de dados, ser capaz de gerar novos dados com as mesmas características com os quais ela ela foi treinada.

Durante o treinamento da rede *GAN*, cada parte é trabalhada em um momento separado. Inicialmente, é feito o treinamento apenas da rede discriminadora. Como é ilustrado na Figura 3, são fornecidos os conjuntos de dados reais e os conjuntos de dados gerados pela rede geradora e, neste momento, a rede discriminadora tenta prever corretamente qual dado foi gerado e qual dado é real. Assim, executando seus ajustes (*backpropagation*) de acordo com seus acertos e erros.

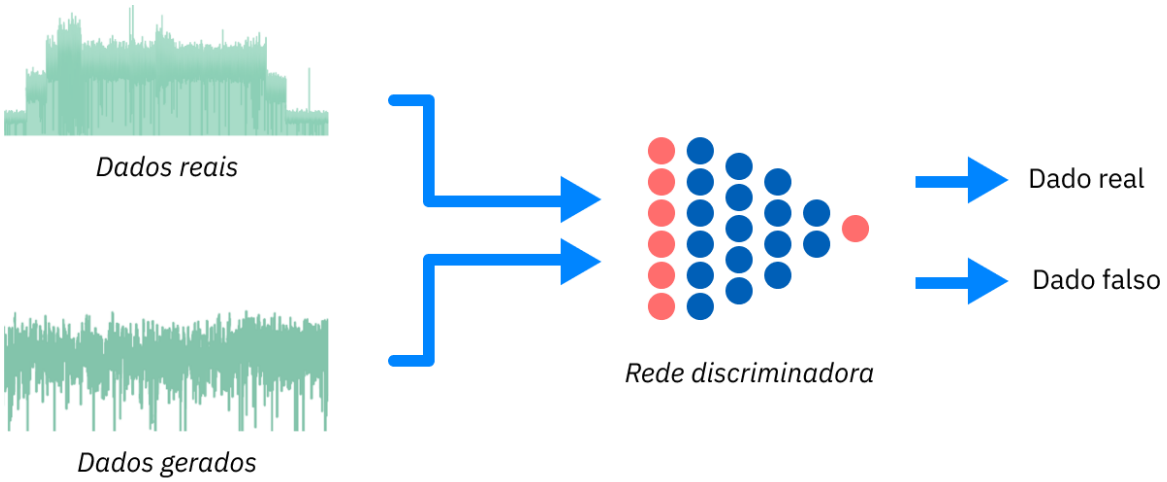


Figura 3 – Treinamento da rede discriminadora

Em seguida, é realizado o treinamento apenas da rede geradora: esse processo é demonstrado na Figura 4. A partir de um ruído aleatório, a rede geradora produz algumas amostras e as fornece para a rede discriminadora. Nesta etapa, seu objetivo é fazer com

que a rede discriminadora classifique seus dados gerados erroneamente como reais. Por isso, aqui os rótulos são invertidos: o erro da rede discriminadora indica o acerto da rede geradora. Dessa forma, de acordo com as classificações que seus dados tiverem, a rede geradora executará seus ajustes para melhorar sua performance e produzir cada vez mais dados semelhantes com os reais.

Figura 4 – Treinamento da rede geradora

Após isso, esse ciclo de treinamento é reiniciado, voltando novamente o treinamento da rede discriminadora. Durante esses passos, é importante notar como a rede generativa não conhece os dados reais em nenhum momento, aprendendo apenas a partir do erro (*loss*) com a rede discriminadora [58, 60].

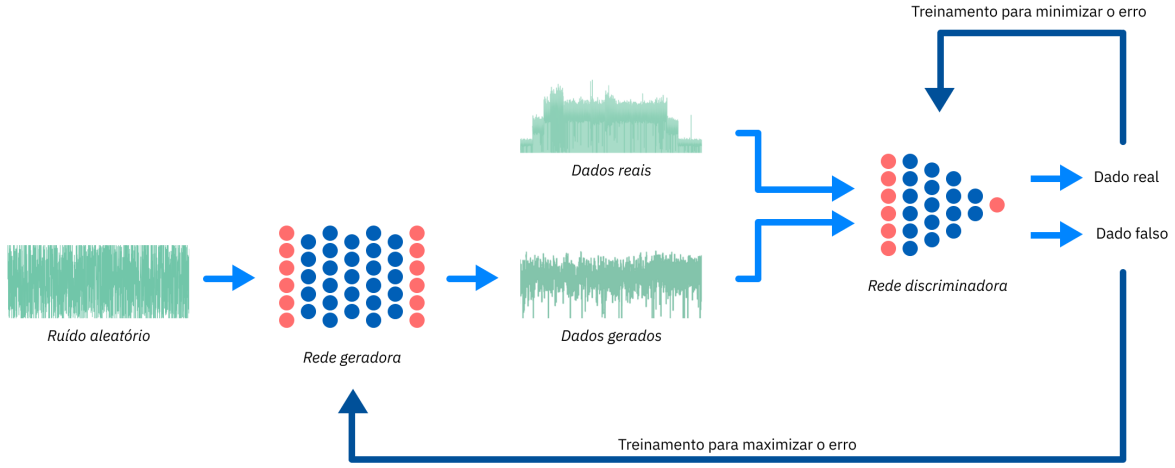


Figura 5 – Estrutura de uma rede *GAN*

De forma completa, conforme mostrado na Figura 5, é possível afirmar que a *GAN* funciona como um mecanismo de *minimax*. Onde a rede geradora tenta superar a rede discriminadora e vice-versa. Os passos do treinamento, conforme ilustrados, acontecem da seguinte forma:

- A rede geradora recebe como entrada um ruído e, a partir dele, tenta gerar algo que imite um dado real.
- A rede discriminadora, por sua vez, recebe como entrada os dados reais e os dados gerados pela outra rede e tenta distinguir entre eles.

- Dessa forma, quando treinamos a rede geradora, buscamos maximizar o erro da rede: significa que tentamos produzir dados que pareçam reais o suficiente para enganar a rede discriminadora com sucesso, e com isso, aumentamos o *erro* (loss) da rede.
- Entretanto, quando treinamos a rede discriminadora, buscamos minimizar o erro, e assim, distinguir corretamente entre as entradas fornecidas.

Sendo uma arquitetura aberta que permite diversas implementações, vários modelos tem sido propostos desde sua apresentação inicial em 2014 [58] para aprimorar e contornar seus problemas originais [61].

Os problemas mais conhecidos das arquiteturas *GAN* incluem a não convergência, quando a rede apenas oscila entre resultados intermediários sem obter uma definição final; a volatilidade do treinamento, quando a rede geradora ou discriminadora se torna muito superior a outra e acabam se travando ou se limitando demais a estrutura dos dados treinados (*overfitting*); o desaparecimento do gradiente, quando a rede geradora acaba não conseguindo aprender o modelo devido as poucas informações retornadas pela rede discriminadora, entre outros.

Vale ressaltar que enquanto algumas problemas como o *overfitting* podem ser mais diretamente solucionados, como por exemplo, com o uso de camadas de *Dropout* que aleatoriamente anulam valores passados entre as camadas. Outros, acabam sendo inerentes a estrutura da própria rede e bem mais difíceis de solucionar, fatores estes que levaram ao desenvolvimento de diversas variantes das redes *GAN*.

Na área de detecção de anomalias em específico, diversas variações têm incrementalmente aprimorado e trabalhado com os problemas conhecidos como *GANomaly* [62], *AnoGAN* [63] e *EGBAD* (*Efficient GAN-Based Anomaly Detection*) [64].

Trabalhos recentes como o apresentado por Novaes [14] têm demonstrado sucesso em combinar o uso da *GAN* para obter melhores resultados na detecção de anomalias de rede. Em especial, motivados por atacar um problema específico que tem crescido ultimamente: anomalias construídas com características específicas para enganar redes neurais de detecção [65, 66]. Nesta abordagem, as redes discriminadoras são utilizadas para a detecção de anomalias e, juntamente com a iterações realizadas com a rede geradora, espera-se que ela aprenda a identificar maiores variações e obter uma melhor performance. Como no caso demonstrado, detectando ataques que outrora não seriam detectados por serem precisamente construídos para enganá-la.

3.3.2 Redes Autoencoder

As redes *Autoencoder* foram propostas pela primeira vez 1986 [67] e possuem uma proposta simples: reconstruir a sua entrada. Este objetivo que a princípio parece trivial,

torna-se um problema complicado mas com propriedades igualmente poderosas quando não se permite um mapeamento identidade entre as partes. Ou seja, não se é permitido que a rede simplesmente entregue sua entrada sem efetuar nenhum processamento.

Para que isso seja alcançado, uma das possibilidades é que a rede possua camadas ocultas dimensionalmente menores (*bottleneck*), forçando-a assim a aprender uma representação interna dos dados treinados. Esta representação interna é geralmente o objetivo final da aplicação desta rede [68, 69].

Estruturalmente, os *Autoencoders* podem ser subdivididos em duas partes, uma primeira responsável por codificar o dado para uma representação interna, geralmente, reduzindo sua dimensionalidade. Uma segunda parte, responsável por decodificar o dado de sua representação interna de volta para sua forma real [70].

Nas figuras 6 e 7, podemos observar o comportamento que é esperado da rede ao se apresentar diferentes dados para ela, assim como sua estrutura: enquanto o codificador reduz a dimensionalidade de cada camada até o ponto de *bottleneck*, o decodificador progressivamente aumenta a dimensionalidade de cada uma até chegar na mesma quantidade que os dados possuíam na entrada.

Usualmente, é comum que se faça a diminuição e o aumento da dimensionalidade de forma espelhada. Apesar de nos exemplos dados a rede realizar a mudança de dimensionalidade de um em um (*Stacked Autoencoder*), é mais comum que a utilização de Autoencoders com dados mais complexos realizam cortes maiores. Por exemplo, enquanto as rede ilustradas levam as dimensionalidades de 6 para 5, 4, 3, 4, 5 e 6, uma rede com dados de maior dimensionalidade, como a título de exemplo, uma com 54 atributos poderia levar as dimensionalidade de 54 para 27, 13, 6, 13, 27 e 54.

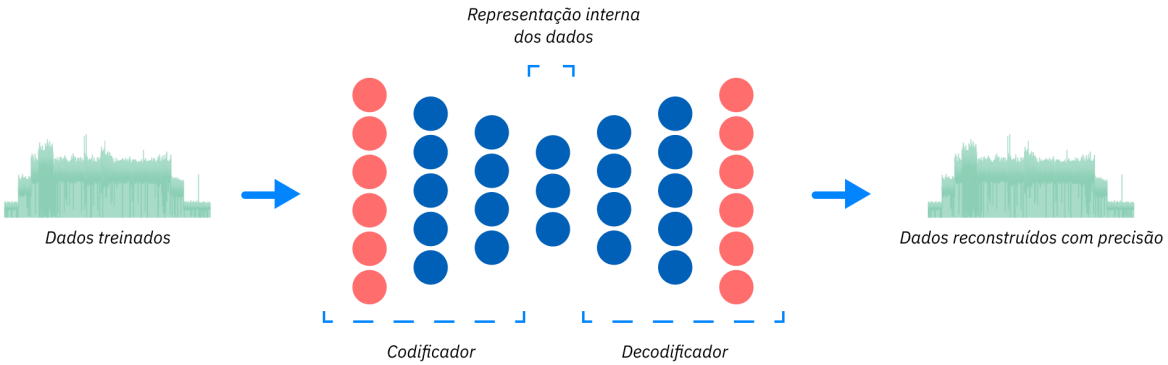


Figura 6 – Funcionamento do *Autoencoder* quando submetido a um dado treinado

Na Figura 6, quando a rede é apresentada a um padrão de dados para o qual ela previamente foi treinada, é esperado que ela reconstrua com a menor distorção possível a amostra que lhe foi dada de entrada.

Já na Figura 7, quando a rede é apresentada a um padrão de dados para o qual ela

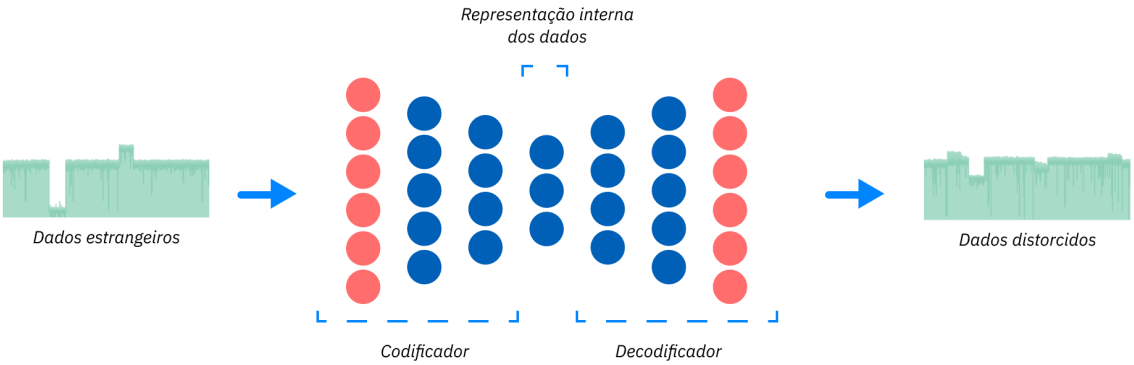


Figura 7 – Funcionamento do *Autoencoder* quando submetido a um dado não treinado

não foi previamente treinada, ou seja, uma anomalia, é esperado que ela produza dados distorcidos. Como demonstrado, o contorno da amostra ao final da rede se mostra bem diferente de quando ele era na entrada.

A partir disto, ao utilizar individualmente suas partes, podemos olhar para o *Autoencoder* como uma rede discriminadora utilizada para executar classificações. Como também, uma possível geradora no caso dos *Variational Autoencoders* [71].

Com a detecção de anomalias em mente, buscamos utilizá-lo diretamente como rede discriminadora. Ao treiná-lo com um determinado dado e/ou padrão, ele aprenderá a representar internamente apenas este tipo de dado. Assim, espera-se que ao fornecer uma entrada diferente da qual ele foi treinada, como por exemplo uma anomalia, a rede falhe em reconstruir com precisão seu dado de entrada. Dessa forma, denotando algo fora de seu padrão conhecido [72, 73].

4 SISTEMAS DE DETECÇÃO DE INTRUSÃO

Sistemas de detecção de intrusão (*IDS*), são ferramentas de defesa automatizadas que detectam atividade maliciosa em um computador ou rede. Elas podem ser caracterizadas por 4 elementos principais: os recursos a serem protegidos, os modelos para identificar o comportamento normal desses recursos, as técnicas para comparar as atividades que estão acontecendo com o comportamento normal e a identificação do que será considerado como anômalo [74].

São sistemas que idealmente são utilizados como uma segunda linha de defesa, apenas entrando em ação quando uma sistema mais rígido, como exemplo, um *firewall* ou outra medida mais básica não é capaz de impedir o ataque. Dado que, a construção destes sistemas são extremamente complexos, dado o seu objetivo excepcionalmente difícil que é a sua tarefa de identificar anormalidades. Motivo pelo qual, Fernandes et al. [27] afirma que por sua vez, estes sistemas não sequer detectam intrusões, mas sim, apenas coletam evidências delas.

4.1 Classificações e tipos

Os IDS podem ser classificados tanto de acordo com o sistema em que são implantados: dizendo com relação a sua fonte de dados com que irá trabalhar, quanto com a técnica de detecção utilizada: de acordo com o mecanismo principal utilizado para identificar comportamentos anômalos [25, 75].

Na Figura 8, é possível visualizar um panorama dos tipos e classificações de *IDS* mais comuns. Inicialmente, entre *NIDS* e *HIDS* de acordo com a origem dos dados utilizada pelo sistema. Segundamente, entre os baseados em anomalia e assinatura, de acordo com o algoritmo propriamente implementado para execução. Ainda nesta imagem, é possível observar a grande quantidade de técnicas utilizadas, especialmente com relação a área de *Machine Learning*.

Fonte: Adaptação própria de [76]

Figura 8 – Mapa de técnicas aplicadas a sistemas de detecção de intrusão

4.1.1 De acordo com a fonte dos dados

Em sua principal distinção, os *IDS* podem ser classificados em *HIDS*, *NIDS* ou *Hybrid* de acordo com a plataforma ou dados que estão à sua disposição [77]:

- *Host-based IDS* (*HIDS*) são projetados para operar em computadores específicos, realizando o monitoramento e análise apenas do sistema em que ele está sendo executado, sem observar a rede ou suas interfaces externas. Permitindo assim, a análise direta de acessos suspeitos a arquivos, chamadas de sistema, entre outros processos e comandos locais.

- *Network-based IDS* (*NIDS*) monitoram e analisam o tráfego de uma rede com o objetivo de proteger todos conectados a ela. Diferentemente dos *HIDS* que possuem acesso direto ao sistema operacional e conseguem verificar diretamente se houve ou não uma intrusão, os *NIDS* usam a premissa de que, como intrusões tendem a gerar padrões irregulares de tráfego e dados na camada de rede, esses mesmos podem ser diretamente detectados e suprimidos.

- *Hybrid* são sistemas híbridos que combinam as funcionalidades de *HIDS* e *NIDS*.

4.1.2 De acordo com a técnica de detecção

Estes sistemas, ainda, podem ser classificados de acordo com a técnica utilizada para executar a análise dos dados. Embora existam diversas classificações possíveis com novos métodos e técnicas sendo utilizados, de forma mais abrangente eles se agrupam em detecção de assinatura, detecção de anomalia e sistemas híbridos:

- Os sistemas baseados em detecção de assinatura (*misuse detection*) analisam a rede utilizando um conjunto de regras ou assinaturas já conhecidos e pré-determinados de ataques. Assim, dependendo de possuir uma base de dados constantemente atualizada com as assinaturas de todas intrusões conhecidas. Se trata de um método que embora traga uma baixa taxa de falsos alarmes [78], pode facilmente pode ser enganado ao realizar pequenas modificações em ataques já conhecidos ou quaisquer itens não previamente mapeados.

- Por sua vez, os sistemas baseados em anomalias, também conhecidos como baseados em perfil, buscam encontrar um padrão esperado na atividade normal da rede. Uma vez feito essa modelagem e aprendido esse "perfil" que represente a ela, o sistema passa a comparar o tráfego atual com o que se esperaria no padrão e classifica como anomalia ou não com base na quantidade de divergência encontrada entre eles.

  Neste método, se torna perceptível a tendência de maiores falsos alarmes e a necessidade de uma atualização periódica do perfil gerado da rede, tratando-se de algo

extremamente maleável aos hábitos dos usuários dela. Todavia, se trata de um método que se mostra capaz de detectar qualquer tipo de intrusão ou atividade não autorizada, pré-existente ou não, uma vez que se baseia exclusivamente no perfil que constitui a normalidade da rede [27].

- Finalmente, os sistemas híbridos combinam o uso de técnicas de detecção de assinatura com de anomalia, entre outros. Um exemplo de aplicação comum é a criação de sistemas baseados em anomalias que juntamente utilizam assinaturas para que quando uma anomalia seja detectada, ela possa ter sua assinatura comparada e assim, possivelmente mapeada a um ataque já conhecido para se tomar uma ação de resposta mais apropriada.

4.2 Detecção de anomalias

Dentre as técnicas apresentadas na seção anterior, as baseadas em anomalia são as mais usualmente utilizadas em *IDS*, sendo também as mais populares e estudadas há mais de 20 anos no ramo científico [27]. Neste tempo, levando ao desenvolvimento dos mais diversos métodos e modelos de análise com desafios até hoje abertos.

Esse progresso pode ser visualizado na Tabela 4 assim como na Figura 8, sumarizando-se os principais métodos até então estudados e explorados dentro do IDS que utilizam a detecção baseado em anomalias. Além de bem como modelos híbridos que combinam várias ramos apresentados na tabela [79].

Tabela 4 – Sumário de métodos utilizados na detecção de anomalias

Categoria	**Descrição**	**Exemplos**
Computação evolucionária	Compreende o algoritmos inteligentes, geralmente incluindo a inteligência artificial inspirados por mecanismos biológicos	Algoritmos genéticos, sistemas imunológicos artificiais, otimização por enxame de partículas
Teoria da informação	Trabalha com o cálculo de entropias, cruzamento de informações mutuais entre outras ferramentas da teoria da informação para identificar anomalias	Entropia de Shannon, Distância de Kullback-Leibler
Máquinas de estado finito	Modelos matemáticos que definem estados, transições e ações; trabalhando fortemente com analítica	FSM, Markov chain
Classificação	Sistemas que são previamente treinados com dados rotulados e tentam classificar um dado em anômalo ou não	Classificador de Naive Bayes, SVM, Redes neurais artificiais
Clustering	Busca agrupar classes de objetos similares (clusters), identificando assim, dados que estão muito longe de qualquer outro grupo	KNN, K-means
Estatístico	Utiliza modelos estatísticos, usualmente trabalhando com o cálculo de *thresholds* para avaliar mudanças o comportamento da rede	Análise de Wavelets, PCA, Matrix de Covariância

5 TRABALHOS RELACIONADOS

O tópico de detecção de anomalias e intrusões em rede vem sendo alvo de inúmeros estudos pelo Grupo de Pesquisa Orion na UEL desde 2004 [80], sendo eles, uma sólida base de início para a pesquisa deste trabalho. Além dele e de todo material base do grupo, se destacam trabalhos como os apresentados a seguir.

No trabalho de Novaes et al. [14], é chamado a atenção para um problema pouco explorado: o uso de ataques adversariais que buscam especificamente confundir técnicas modernas de detecção de anomalias. Uma abordagem especialmente problemática para redes neurais profundas, uma vez que, estes ataques, utilizam-se de *features* especificamente construídas contra os padrões de anomalia aprendidos por ela. Neste trabalho, é proposto a criação de um sistema de defesa para redes *SDN* baseada na arquitetura *GAN* sob a premissa de igualmente conseguir gerar ataques adversariais e treinar previamente o sistema para isso através dos mecanismos intrínsecos dela. Ao final, o resultado obtido foi superior a todas outras arquiteturas testadas, demonstrando o potencial da arquitetura *GAN* sugerida.

Em Scaranti et al. [2], é proposto a utilização de um sistemas imunológico artificial, denominado AIS-IDS (*Artificial Immune System based IDS*). Utilizando-se do algoritmo de seleção negativa (NSA), o modelo simula o reconhecimento biológico de anticorpos, segregando dados que compõe ele mesmo, para por fim, calcular sua similaridade com a aplicação de lógica *fuzzy* para determinar se tratar de uma anomalia ou não. Em comparação a outros algoritmos como o de *Naive Bayes* e K-NN, o AIS-IDS superou quaisquer outros resultados, alcançando valores de 99.97% na métrica F1.

Na publicação de Chen et al. [69], traz-se a aplicação de um *Autoencoder* Convolucional: uma rede com menos parâmetros e menos treinamento exigido em comparação com *autoencoders* tradicionais. Em comparação com os métodos observados, PCA e *Autoencoder* convencional, foi possível observar a melhora das métricas na curva ROC, na taxa de falsos positivos e de acurácia em quase todos os cenários apresentados, demonstrando com sucesso o método apresentado.

Niyaz et al. [81] demonstra a aplicação de um *Stacked Autoencoder* em um ambiente *SDN* bruto com o controlador para detecção de anomalias com ênfase em DDoS. Neste trabalho, é executada a extração e seleção manual de 34 *features* diferentes de fluxos TCP, 19 de fluxos UDP e 13 de fluxos ICMP, para enfim, serem processados pela rede. Para a execução dos resultados, foi simulado uma rede residencial com 12 usuários conectados, obtendo resultados superiores aos métodos comparados.

Carvalho et al. [7] traz um ecossistema completo para detecção e mitigação de

anomalias. Nele, são propostos a utilização da técnica de Otimização de colônia de formigas para assinatura digital (*ACODS*) para detecção e de um modelo de regressão logística multinomial (*MLR*) para identificação delas. Nesta arquitetura, é proposto um mecanismo que extrai 6 atributos da rede (bits, pacotes, entropia de IP e porta de origem e destino) e define um comportamento de tráfego esperado para a rede (*DSNSF*), utilizando-o para comparar com o estado atual da rede. Sendo identificado alguma anomalia pelo *ACODS*, é feito o reconhecimento do tipo dela pelo *MLR* e, finalmente, tomada as ações necessárias para sua mitigação. Os resultados do trabalho foram comparados com os obtidos por modelos *SVM* e *CKNN* e se mostraram superiores em todas as métricas durante a execução sobre um conjunto de amostras reais de rede, contendo nele, com ataques *DDoS*, *PortScan* e *Flash Crowd*.

No trabalho de Hamamoto et al. [4], é empregado um algoritmo genético para gerar as caracterizações do tráfego de rede (*DSNSF*) onde, posteriormente, com o uso da Lógica *Fuzzy* acompanhada da função Gaussiana de Pertencimento, determinar a anormalidade ou não de determinado tráfego na rede. Para a avaliação dos resultados, foram utilizados dados colectados na Universidade Estadual de Londrina utilizando o protocolo sFlow contra algoritmos como CkNN, SVM, ACODS, atingindo resultados distantemente melhores com uma acurácia de 96.53%.

No artigo de Assis et al. [15], é apresentado um mecanismo de resposta rápida a ataques e que possa ser implementado diretamente no controlador de uma rede SDN, justamente, localização de um dos pontos vulneráveis da arquitetura. Para isso, é utilizado uma rede neural profunda GRU (*Gated Recurrent Units*), executando sobre as *features* previamente extraídas (bits, pacotes, porta de destino e porta de origem, juntamente com um hashing de outras informações quantitativas). Nele, é comparado o uso de outras redes como DNN, CNN, LSTM, SVM, além de outras tradicionais, analisando-se a eficiência e os requisitos de cada um nos conjuntos de dados CICDDoS 2019 e CICDDoS 2018. Ao longo das execuções, embora não obtendo sempre os melhores resultados, a rede proposta demonstrou ter o melhor equilíbrio entre as métricas analisadas, além de uma performance superior as outras quando se analisando apenas as classificações de tráfegos legítimos.

Em Andresini et al. [82], propõe-se uma nova métrica de aprendizado profundo com a utilizações *Autoencoders* juntamente com redes *Triplet* para determinar as seleções finais indicadas pela rede. Diferentemente de outros trabalhos, é efetuado o treinamento de *autoencoders* tanto com amostras normais quanto de ataques, permitindo assim, a classificação na rede Triplet com uma maior precisão dados. Neste trabalho, utilizou-se os datasets KDDCUP99, AAGM17 e CICIDS17, compreendendo ataques de força bruta, *portscan*, *DDoS*, infiltração, entre outros. Em comparação a outros algoritmos como CNN, GAN, e outras combinações de Autoencoder c com CNN e MLP, a arquitetura proposta se mostrou a com maiores valores alcançados nas métricas de acurácia e F1.

Niu et al. [83] apresenta um mecanismo para detecção de anomalias dentro de séries temporais com a utilização de uma GAN construída com redes VAE (*Variational Autoencoders*) e LSTM (*Long short-term memory*). Neste trabalho, é construído uma rede discriminadora e geradora que são constituídos de unidades LSTM, mas aplicadas arquiteturalmente com um VAE. Na pesquisa realizada, é feito a avaliação com dados provenientes de sensores, utilizando-se o dataset KPI e Yahoo. Para a avaliação dos métodos, é proposta a comparação com as redes LSTM-AE, LSTM-VAE e MAD-GAN: o resultado obtido é que a rede LSTM baseada em VAE-GAN proposta consegue não só obter resultados superiores, como também, menores tempos de treinamento.

Zenati et al. [84] traz uma nova abordagem para detecção de anomalias de forma baseada em GAN. Em seu trabalho, é proposto uma arquitetura de aprendizado baseado de forma adversária (*Adversarially Learned Anomaly Detection*). Utilizando uma modificação das arquiteturas de *GANs* bidirecionais, a ideia é que as características a serem aprendidas sejam derivadas adversarialmente. Os resultados da pesquisa foram testados com os conjuntos de dados KDD99, Arrhythmia, CIFAR-10 e SVHN, em comparação com os modelos AnoGAN, SVM, DAGMM (Deep Autoencoding Gaussian Mixture Model), OC-SVM (One Class Support Vector Machines) entre outros que representam o estado da arte. A conclusão das métricas finais indicam que o método proposto possui resultados competitivos, mas se mostra extremamente mais promissor ao escapar de problemas comuns da arquitetura *GAN*.

Por fim, é possível visualizar um resumo dos métodos, conjuntos de dados e algoritmos comparados por cada um dos trabalhos relacionados na Tabela 5.

Tabela 5 – Resumo dos trabalhos relacionados

Autor	**Método proposto**	**Dataset**	**Algoritmos comparados**
Novaes et al. [14]	*GAN*	CiCDDoS 2019	CNN, LSTM, MLP
Scaranti et al. [2]	Sistemas imunológicos artificiais	CiCDDoS 2019; Dataset próprio	Naive Bayes, K-Nearest Neighbors, Local Outlier Factor
Chen et al. [69]	*Autoencoder* convolucional	KDD	PCA, Deep Autoencoder
Niyaz et al. [81]	*Stacked autoencoder*	Dataset próprio	Soft max, MLP
Carvalho et al. [7]	*ACODS*	Dados da Universidade Estadual de Londrina	SVM, CKNN
Hanamoto et al. [4]	Algoritmo genético	Dados da Universidade Estadual de Londrina	CKNN, SVM, ACODS
Assis et al. [15]	*GRU*	CiCDDoS 2019;CoCDDoS 2018	DNN, CNN, LSTM, SVM
Adresini et al. [82]	*Autoencoder* com *Triplet* Network	KDDCUP99; AAGM17; CiCIDS17	CNN, MLP, Deep Autoencoder
Niu et al. [83]	*VAE-GAN* baseada em LSTM	KPI; Yahoo	LSTM-AE, LSTM-VAE, MAD-GAN
Zenati et al. [84]	ALAD (*GAN* Bidirecional)	KDD99; Arrhythmia; CIFAR-10; SVHN	OC-SVM, IF, DSEBM, AnoGan

6 DESENVOLVIMENTO

Conforme demonstrado anteriormente, as capacidades em potencial das redes de aprendizado profundo *GAN* e *Autoencoder* mostram-se suficientemente atrativas para resolução de problemas como delimitação de normalidade e detecção de anomalias. Assim, neste trabalho realizou-se um estudo prático com a implementação destas redes neurais, bem como um experimento de sua possível utilização em conjunto.

6.1 Cenário de dados

Para realização dos estudos, foram propostos a análise e utilização de dois conjuntos de dados disponibilizados pelo Grupo de Redes Orion, da Universidade Estadual de Londrina. Ambos sendo constituídos pela coleta de fluxos em uma rede emulada através de uma *SDN* ao longo de 3 dias, sendo destes, o primeiro com apenas tráfego normal e os dois últimos com ataques *DDoS* e *PortScan* ao longo do dia.

Topologicamente, o conjunto de dados A foi obtido em uma rede com 100 computadores (*hosts*), enquanto o conjunto B, em um universo de 200 computadores. Nos dois casos, sendo todas conexões interligadas através de 6 *switches*.

Na Tabela 6 é possível observar os eventos presentes ao longo dos dias em cada cenário de dados, mostrando a presença de ataques *DDoS* e *PortScan* nos dois dias de ataque com durações e horários variados sem a sobreposição dos mesmos em nenhum momento. Valores estes que também são confirmados na visualização dos dados apresentados na Figura 9.

Tabela 6 – Descrição dos eventos presentes nos conjuntos de dados

		Conjunto de dados A	**Conjunto de dados B**
Dia 1		Tráfego normal	Tráfego normal
Dia 2	DDoS	09:35 - 10:45	09:15 - 10:20
	PortScan	14:30 - 15:30	15:35 - 16:40
Dia 3	DDoS	10:32 - 11:45	14:20 - 15:20
	PortScan	16:45 - 17:45	16:12 - 18:26

Neste momento inicial, os dados brutos presentes de cada um dos fluxos capturados detalham seu horário de início, endereço IP e porta de destino, endereço IP e porta de origem, quantidade de pacotes e *bytes*, assim como sua duração.

6.1.1 Extração de *features* e modelo

Em posse destes dados, é necessário que se haja um pré-processamento para que se tornem computáveis pelas redes neurais. Neste ponto, não era possível trabalhar diretamente com o conjunto, dado que os fluxos eram individuais e com durações variáveis. Dessa forma, o primeiro passo necessário era a transformação deles para um conjunto de dados que sumarizasse o estado da rede a cada determinado momento.

Para tal, realizou-se uma reconstituição da captura, de forma a reconstruir o estado da rede a cada segundo do dia. Transformando os fluxos em 86400 janelas de captura que agregavam as estatísticas do que acontecia na rede em um determinado segundo específico.

Contudo, nesta agregação, enquanto dados propriamente numéricos como quantidades de pacotes e *bytes* podiam ser diretamente operáveis, valores como endereços de IP e portas representam dados nominais e, em face disso, ainda precisariam ser transformados para que pudessem ser quantitativamente analisados.

Dessa maneira, foi utilizada a Entropia de Shannon [85] para transformar os endereços de IP e portas em valores que representassem a concentração ou dispersão de sua ocorrência. Em outras palavras, passando a trabalhar com suas entropias. Técnica comumente utilizada na Teoria da informação para o tratamento de dados com propriedades similares.

A fórmula utilizada para este cálculo é demonstrada abaixo, onde $H(x)$ corresponde a entropia de x, S a soma de todas as ocorrências presentes no intervalo analisado, x_i a ocorrência analisada, $\frac{x_i}{S}$ a probabilidade desse evento acontecer e N a quantidade de eventos possíveis.

$$H(x) = -\sum_{i=1}^{N}(\frac{x_i}{S})\log_2(\frac{x_i}{S})$$

Com os valores de entropia dos atributos desejados, passa-se a trabalhar com um valor numérico que indica o nível de desordem no período analisado. Por exemplo, durante um ataque de *DDoS*, é comum que o tráfego da rede apresente uma concentração elevada de um mesmo IP e porta de destino já que há uma quantidade massiva de dados direcionado ao mesmo alvo: dessa forma, seu valor de entropia será menor que o normal.

Por outro lado, durante um ataque *PortScan*, onde ocorre uma quantidade de requisições para diferentes portas muito maior que o normal, é comum que entropia tenha um valor elevado devido a essa desordem. Esses comportamentos podem ser observados mais explicitamente na Figura 9.

Por fim, definindo um modelo com 6 atributos (*features*) finais: *bytes*, pacotes, entropia de IP de origem, entropia de IP de destino, entropia de porta de origem e entropia de porta de destino, conforme ilustrado abaixo.

Quantidade de Bytes	Quantidade de Pacotes	Entropia de IP de Destino	Entropia de Porta de Destino	Entropia de IP de Origem	Entropia de Porta de Origem

Em suma, transformando cada dia dos conjuntos de dados em 86400 amostragens de cada uma das seis dimensões definidas, sendo uma amostra para cada segundo do dia. A visualização do resultado obtido após esse processamento é demonstrado na Figura 9. Nela, são demonstrados o comportamento das 6 dimensões previamente definidos ao longo das 24 horas do Dia 2 do cenário A.

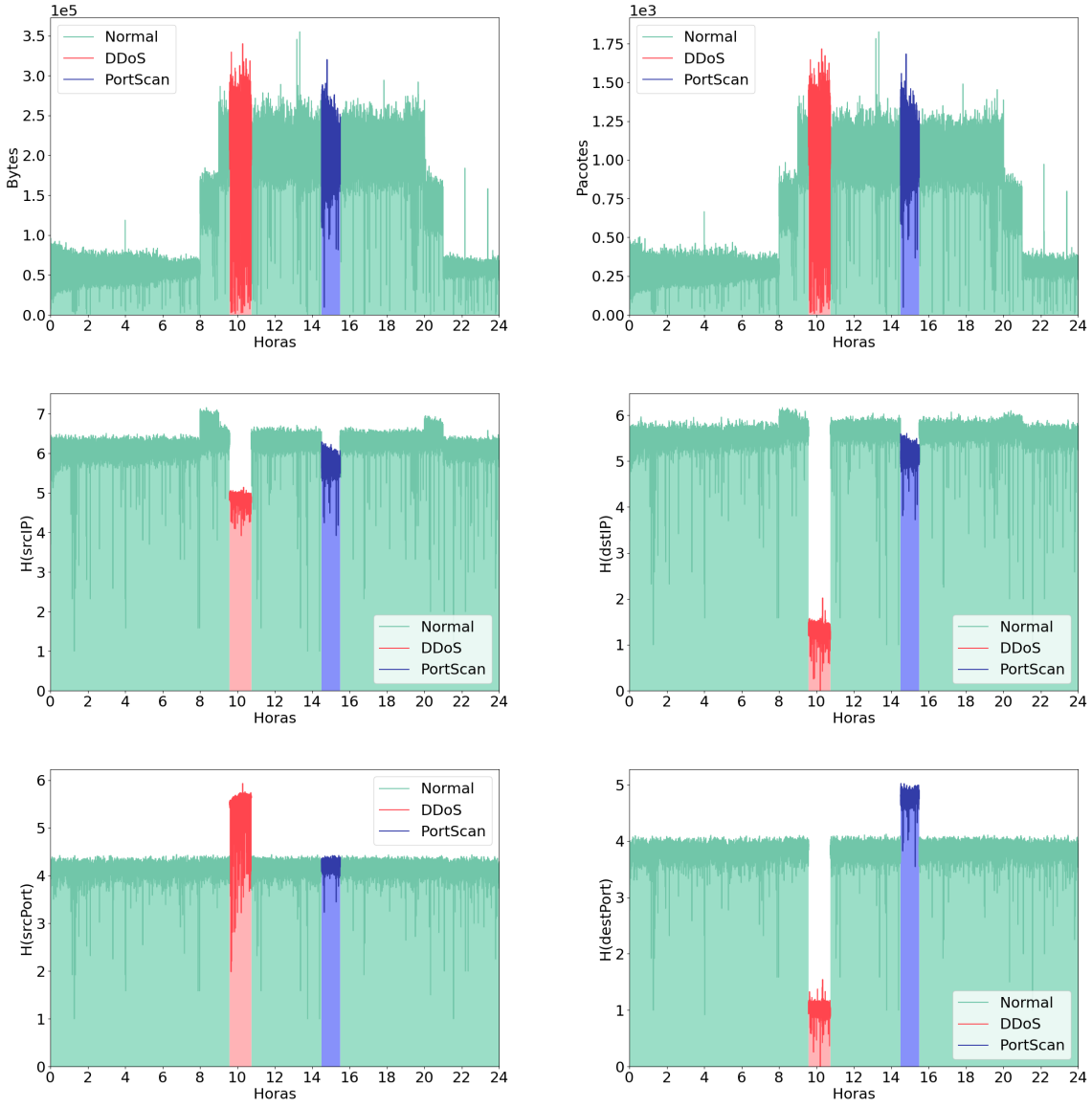


Figura 9 – Gráfico das dimensões ao longo do tempo no Dia 2 do Conjunto de Dados A

Este dia possui ataques *DDoS* no período da manhã e ataques de *PortScan* no período da tarde. Assim, é possível visualizar o impacto que cada um tem sob cada atributo. Especialmente, na entropia de porta de destino (*H(destPort)*), que se mostra a mais alterada em ambos os ataques, funcionando como um importante indicador não só

da presença de uma intrusão, como possivelmente de qual ataque em específico está sendo realizado, por se alterar respectivamente de forma negativa e positiva para cada um deles.

6.1.2 Normalização

Como preparação final para o treinamento das redes neurais, os dados foram normalizados entre 0 e 1 a partir de seus mínimos e máximos registrados no dia 1 conforme a Equação 6.1, implementada pela função *MinMaxScaler* da biblioteca *SKLearn* [86]. Tendo X_i como um dado pertencente ao conjunto X, $max()$ e $min()$ funções que retornam o maior e o menor valor do conjunto.

$$X_{escalado} = (X_i - min(X))/(max(X) - min(X)) \tag{6.1}$$

Ainda, com o objetivo de reduzir a quantidade de ruídos presentes no treinamento, os dados foram reagrupados em janelas com a média dos valores registrados em espaços de 5 segundos. Reduzindo de 86400 amostras para 17280, e consequentemente, o tempo de treinamento e complexidade.

6.1.3 Avaliação

Como em um cenário real, a construção de quaisquer métricas necessárias assim como o treinamento das redes neurais serão feitos apenas com base no dia 1 de cada conjunto, representando assim, a normalidade esperada do tráfego na rede. Para as avaliações de sucesso dos sistemas como os cálculos de acurácia, precisão, revocação e *F1*, serão utilizados os dias 2 e 3 representando dados nunca antes vistos pelo modelo.

Tabela 7 – Fórmula das métricas utilizadas

Acurácia	$\frac{VP + VN}{VP + VN + FP + VN}$
Precisão	$\frac{VP}{VP + FP}$
Revocação	$\frac{VP}{VP + FN}$
F1	$2 * \frac{Precisão * Revocação}{Precisão + Revocação}$

Os cálculos dessas métricas são descritos na Tabela 7, utilizando-se dos valores da matriz confusão resultante, onde VP corresponde aos verdadeiros positivos, indicando ataques que foram apontados pelo modelo como ataques; FN aos falsos negativos, quando o modelo classifica incorretamente um ataque como tráfego normal; FP aos falsos positivos, sendo os casos em que o modelo classifica incorretamente um tráfego normal como ataque e VN aos verdadeiros negativos, indicando dados normais que foram corretamente apontados como tais.

Tabela 8 – Descrição das métricas

Acurácia	Indica quantas classificações no total foram feitas corretamente
Precisão	Indica dos momentos em que o modelo indicou que havia um ataque, quantos eram de fato um ataque. Uma precisão baixa significa que o modelo está acusando ataques em momentos que não existem.
Revocação	Indica dos ataques que foram realizados, quantos o modelo conseguiu identificar que havia um ataque. Uma revocação baixa significa que o modelo está deixando de acusar ataques, classificando-os como normais.
F1	Representa uma média harmônica entre a precisão e a revocação

Durante as otimizações dos modelos neste trabalho, será buscado o aperfeiçoamento da métrica *F1* por representar uma visão geral dos resultados gerados pela rede. Dado que embora acurácia alta seja importante, ela pode ser facilmente enganadora; um exemplo seria o caso onde o modelo classificasse todas entradas como tráfego normal: neste cenário, ele ainda obteria uma acurácia em torno de 90% nos conjuntos de dados testados. Por outro lado, uma análise conjunta da precisão e da revocação trariam uma visão geral tanto das classificações de ataques quanto de dados normais. Assim, justificando a escolha da métrica *F1*.

Na Tabela 8, é apresentada uma breve explicação do que significa cada métrica e o que seu valor pode indicar a respeito da performance do modelo avaliado, mostrando efetivamente a correlação que se busca obter ao se basear na métrica *F1*.

6.2 *Autoencoder*

Para construção da rede, optou-se pela criação de um *stacked autoencoder*. Sua estrutura ficou composta por 6 camadas densas completamente interligadas e é demonstrada na Tabela 9. Como é ilustrado, a camada de entrada inicialmente se conecta com uma camada densa de dimensão 5 com um ativador $ReLU$ e após percorrer todas as camadas ocultas, se encerra com outra camada de saída densa de dimensão 6, igualmente a entrada. Nesta última etapa, o ativador *Sigmoid* é propositalmente escolhido para garantir que a saída fique no intervalo previamente normalizado de 0 a 1.

Para o treinamento, utilizou-se o otimizador *Adam* e a métrica de perda *MSE* (*Mean Squared Error*). Sendo estas, configurações geralmente sugeridas no desenvolvimento de redes *Autoencoder*.

Durante a avaliação da rede, diversas configurações de camadas e hiperparâmetros foram testadas e continuamente analisadas até se concluir neste formado apresentado. Nos testes realizados, enquanto redes com menos camadas apresentavam dificuldades para reproduzir a entrada e resultavam em atributos com valores zerados, redes com mais ca-

Tabela 9 – Descrição das camadas do Autoencoder

Autoencoder			
	Tipo	**Dimensão**	**Ativador**
Camada de entrada		6	
	Densa	5	ReLU
	Densa	4	ReLU
	Densa	3	ReLU
	Densa	4	ReLU
	Densa	5	ReLU
Camada de saída	Densa	6	Sigmoid

madas tendiam a realizar um *overfitting* dos dados apresentados e falhavam em reproduzir dados com as mais ligeiras divergências que eram esperadas.

No tangente as quantidades de épocas e tamanhos de *batch*, observou-se um grande impacto em função dos tamanhos escolhidos, mas pouca variação em função das quantidades de épocas, gerando resultados que rapidamente se estabilizavam. Ao final, os melhores resultados foram encontrados com a utilização de 100 épocas e *batchs* com tamanho de 64.

6.2.1 *Threshold*

Com a rede neural construída, espera-se que o *Autoencoder* retorne uma reconstrução dos dados que lhe foram fornecidos. Idealmente, funcionando como na Figura 10, onde dado um tráfego normal que corresponda ao padrão que ele foi treinado, resulte em uma reconstrução de alta fidelidade da entrada. Mas dado uma entrada anômala, como um ataque, resulte em uma reconstrução com grandes divergências de sua admissão. Ainda nesta figura, é possível observar o número de neurônios em cada camada oculta, representados pelos círculos, conforme mais detalhado na Tabela 9.

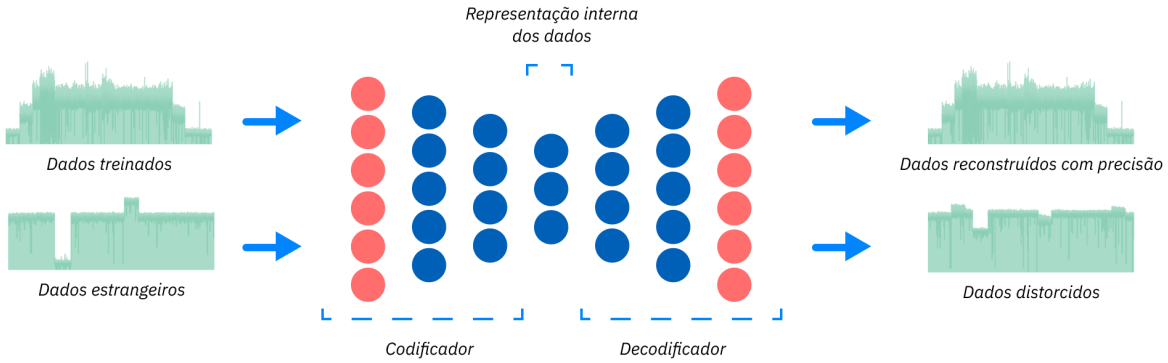


Figura 10 – Estrutura do *Autoencoder* desenvolvido

Desta forma, após a execução do *Autoencoder*, realizamos o cálculo do *MSE* e obtemos um valor que indica a discrepância entre a entrada fornecida e sua reconstrução.

Cabendo agora ao modelo, definir um *threshold*: um limite que defina a partir de qual valor será considerado suficientemente divergente para ser anômalo ou não.

Após a análise de diferentes métodos, optou-se pela inequalidade de Bienaymé-Chebyshev para o cálculo deste limite. Um método já utilizado anteriormente para detecção de anomalias em situações onde a distribuição dos dados é desconhecida [87, 3, 88].

Sua inequação é representada em 6.2, onde X representa uma variável aleatória, μ a média e σ o desvio padrão. Desta forma, temos que valores mais extremos que K desvios padrões da média, serão considerados anômalos.

$$P(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2} \tag{6.2}$$

Valor este que será pré-calculado ainda na fase de treinamento, assim, tomando como regra os valores obtidos no dia 1 para definir a média e o desvio padrão esperados.

Na execução específica deste trabalho, foi trabalhado com o valor de $k = 21.1$, assumindo a probabilidade de encontrar uma anomalia ser menor que 1% e uma distribuição de dados unimodal [87].

$$\mu + 21.1 * \sigma \tag{6.3}$$

Assim, temos que o valor obtido para o limiar é descrito na equação 6.3. Sendo μ, a média e σ o desvio padrão dos valores obtidos pelo cálculo do *MSE* ao longo do conjunto inteiro de treinamento.

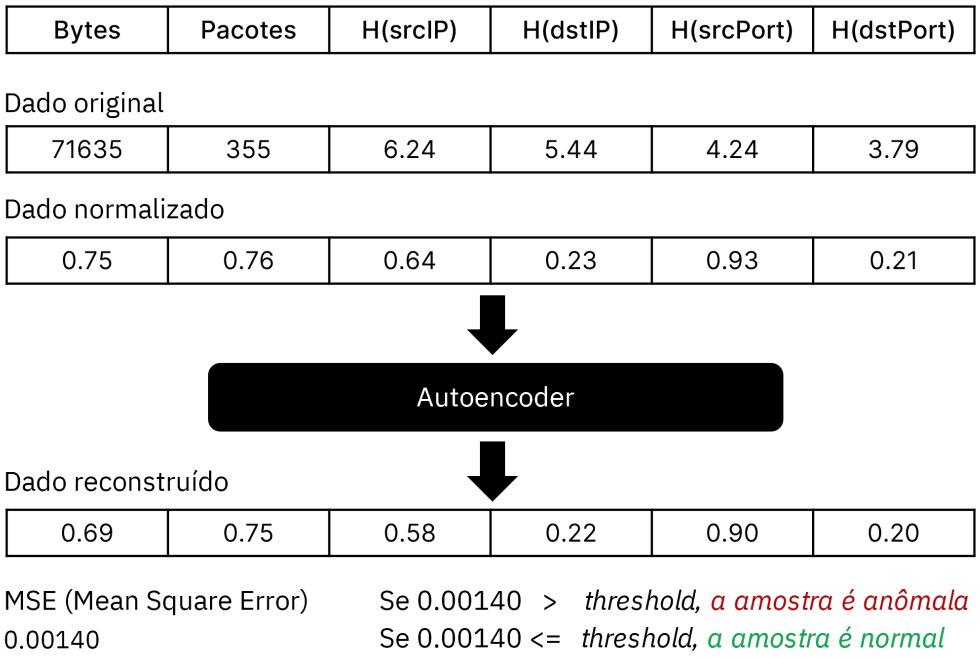


Figura 11 – Representação do processamento de uma amostra pelo *Autoencoder*

Em síntese, na Figura 11, é demonstrado o processamento de um determinado dado ao longo do *Autoencoder*. Nele, a amostra é inicialmente normalizada e, em seguida, fornecida ao *Autoencoder*. Após isso, a rede tenta reconstruir o dado e, por fim, é calculado a diferença (*MSE*) entre o dado de entrada da rede e o dado reconstruído por ela na saída. Para determinar se tal amostra é uma anomalia, o valor obtido pelo cálculo do *MSE* é comparado ao limiar previamente estabelecido: se o valor encontrado for maior que o *threshold*, a amostra será considerada anômala.

6.2.2 Resultados

Aplicando-se a execução do *threshold* encontrado aos dados reconstruídos pela rede *Autoencoder*, obteve-se os seguintes resultados conforme descritos na Tabela 10.

Tabela 10 – Resultados obtidos com o *Autoencoder*

	Conjunto de dados A		Conjunto de dados B	
	Dia 2	Dia 3	Dia 2	Dia 3
Acurácia	99.82%	99.97%	99.52%	92.56%
Precisão	99.61%	99.75%	99.73%	99.90%
Revocação	98.39%	100%	95%	44.88%
F1	99%	99.87%	97.30%	61.94%

No cenário de dados do conjunto A, pode-se observar como a rede consegue entregar uma performance sustentável de forma excepcional ao longo dos 2 dias de testes, levemente deixando de caracterizar alguns ataques no dia 2 com uma revocação de 98.39%. Todavia, começa a apresentar dificuldades no cenário B, tendo uma performance aceitável no dia 2, mas apresentando maiores dificuldades no dia 3, ao detectar apenas 44.88% dos ataques deste dia.

Essa discrepância é, em parte, justificada pela diferença do comportamento da rede entre os dois conjuntos apesar de terem sido gerados de formas similares. Uma inspeção mais detalhada mostra uma maior inconformidade ao longo dos dados trazidos pelo conjunto B do que pelo conjunto A, dificultando o cálculo de um *threshold* que acomode confortavelmente todo o conjunto de dias apresentado.

Ainda assim, ressalta-se a hipótese de que a dificuldade, encontrada pela rede durante as avaliações, conglomera-se em torno da definição do *threshold*. Questão, essa, demonstrada através de testes individuais que indicavam que para todos os dias e cenários, existia pelo menos um valor possível de *threshold* que alcançava 100% de acurácia. Indicando de certa forma, um êxito por parte da rede neural ao conseguir sob condições perfeitas, criar um distinção completa entre o tráfego anômalo ou normal.

6.3 *GAN*

Uma rede *GAN*, assim como descrita anteriormente, traduz-se em duas redes neurais que são colocadas de forma adversária durante o treinamento. Na Figura 12, é apresentada a estrutura da rede desenvolvida neste trabalho, assim como suas camadas e dimensionalidades.

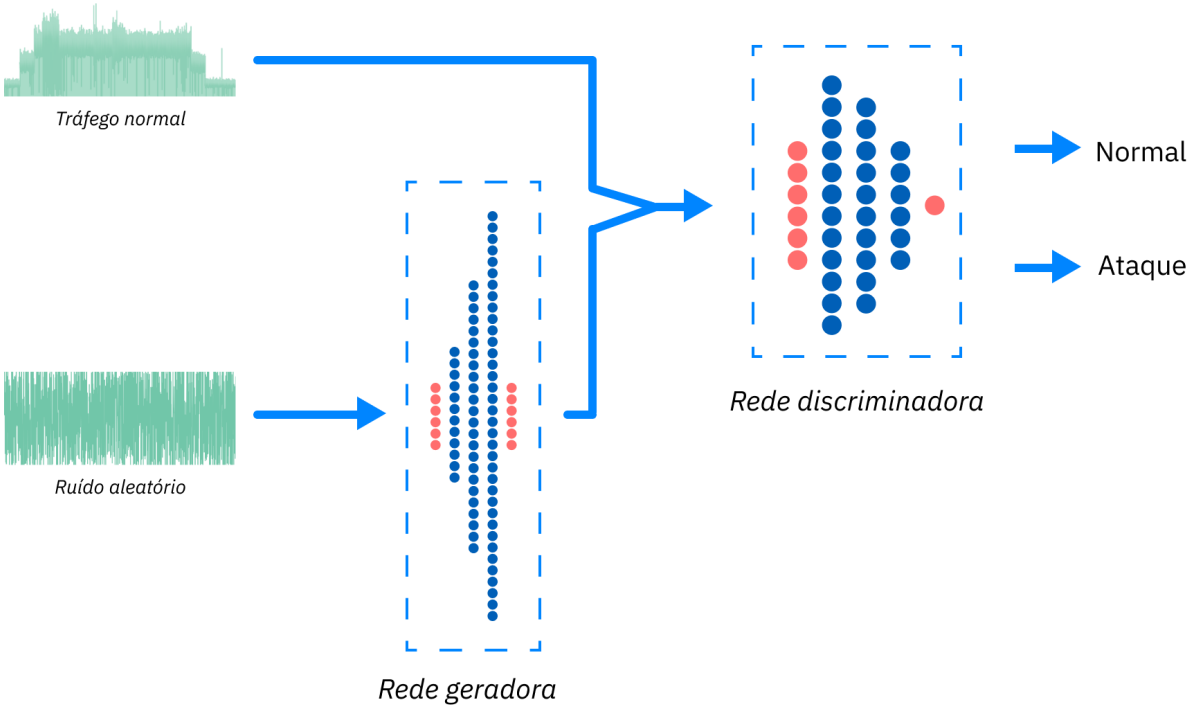


Figura 12 – Estrutura da *GAN* desenvolvida

Para a construção da rede geradora, apresentada na Tabela 11, foi feita a modelagem de uma *MLP* responsável por expandir os 6 ruídos de entrada e transformá-los em um dado com 6 dimensões (*bytes*, pacotes, IP de destino, IP de origem, porta de destino, porta de destino, assim como nos dados provenientes do conjunto de dados).

Tabela 11 – Descrição das camadas da rede geradora da *GAN*

Rede geradora - GAN			
	Tipo	**Dimensão**	**Ativador**
Camada de entrada		6	
	Densa	6	ReLU
	Densa	12	ReLU
	Densa	24	ReLU
	Densa	36	ReLU
Camada de saída	Densa	6	Sigmoid

Como mostrado na Tabela 11, a camada de entrada dos dados é inicialmente conectada a uma camada densa de 6 dimensões com o ativador *ReLU* e percorre a rede

até a saída, onde uma camada densa de 6 dimensões com o ativador *Sigmoid* entregará os dados gerados.

Para a rede discriminadora, apresentada na Tabela 12, é feita também a construção de uma *MLP*, mas agora, responsável por receber os 6 atributos que descrevem o tráfego da rede em um determinado momento e indicar a probabilidade dele ser um ataque. Assim, tendo como sua saída um único valor com o ativador *Sigmoid*.

Ainda na Tabela 12, é possível observar como, nesta rede, são intercaladas camadas densas com camadas de *Dropout* para evitar o *overfitting*. Mas que, diferentemente das outras, termina com uma camada com uma única dimensão com valores entre 0 e 1 (*Sigmoid*), determinando a probabilidade de um ataque.

Tabela 12 – Descrição das camadas da rede discriminadora da GAN

Rede discriminadora - GAN			
	Tipo	**Dimensão**	**Ativador**
Camada de entrada		6	
	Densa	12	ReLU
	Dropout		20%
	Densa	10	ReLU
	Dropout		20%
	Densa	6	ReLU
	Dropout		20%
Camada de saída	Densa	1	Sigmoid

Em ambas as redes, foi utilizado o otimizador Adam e a métrica de perda *Binary Crossentropy*, uma vez que agora a própria rede tem diretamente como saída uma classificação binária entre anomalia e não anomalia, sem executar processos externos como o Chebyshev.

6.3.1 Resultados

Com o objetivo de que se pudesse melhor avaliar os resultados, frente a natureza instável e sensível já conhecida das arquiteturas *GAN*, realizou-se um *Grid Search*. Assim, de forma a contemplar o potencial que a rede pudesse oferecer utilizando-se diferentes valores para quantidade de épocas e tamanho de *batch*.

Nas baterias de teste, considerou-se a execução de até 4096 épocas com os tamanhos de *batch* 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192 e 16386 para cada cenário, incluindo os dois dias de teste além do de treinamento. Ao final da execução, o melhor resultado da métrica F1 combinado ao longo de todos os cenários foi obtido na execução de 1770 épocas e *batchs* com tamanho de 2048.

Tabela 13 – Resultados obtidos com a GAN em comparação com o Autoencoder

	Conjunto de dados A				Conjunto de dados B			
	Dia 2		Dia 3		Dia 2		Dia 3	
Acurácia	99.97%	+0.15%	99.97%	–	99.87%	+0.35%	98.02%	+5.45%
Precisão	100%	+0.38%	100%	+0.25%	100%	+0.26%	97.88%	-2.02%
Revocação	99.96%	+1.57%	99.97%	-0.02%	99.86%	+4.86%	99.88%	+54.99%
F1	99.98%	+0.98%	99.98%	+0.11%	99.93%	+2.62%	98.87%	+36.92%

Na Tabela 13, é possível conferir os resultados, onde, em comparação com o *Autoencoder* previamente apresentado, é perceptível a evolução das métricas avaliadas. Em excepcional, no dia 3 do Cenário B que agora passa a ter resultados no mesmo patamar que dos demais dias processados, chegando a ter um aumento de 36.92% na métrica F1 no terceiro dia.

6.4 *GAN* com *Autoencoder*

Embora a rede *GAN* construída tenha apresentado resultados satisfatórios, seus custos computacionais tanto de treinamento como de aperfeiçoamento de seus hiperparâmetros, ainda combinados à sua volatilidade, fazem um contrapeso considerável não somente ao *Autoencoder*, mas também a outros sistemas e tipos de redes neurais.

Por conseguinte, surgiu-se a hipótese de modificar a *GAN* de forma que ela pudesse ser mais rapidamente treinada e ter uma maior consistência de resultados ao longo das execuções. A ideia, apresentada na Figura 13, é a adição do *Autoencoder* previamente desenvolvido.

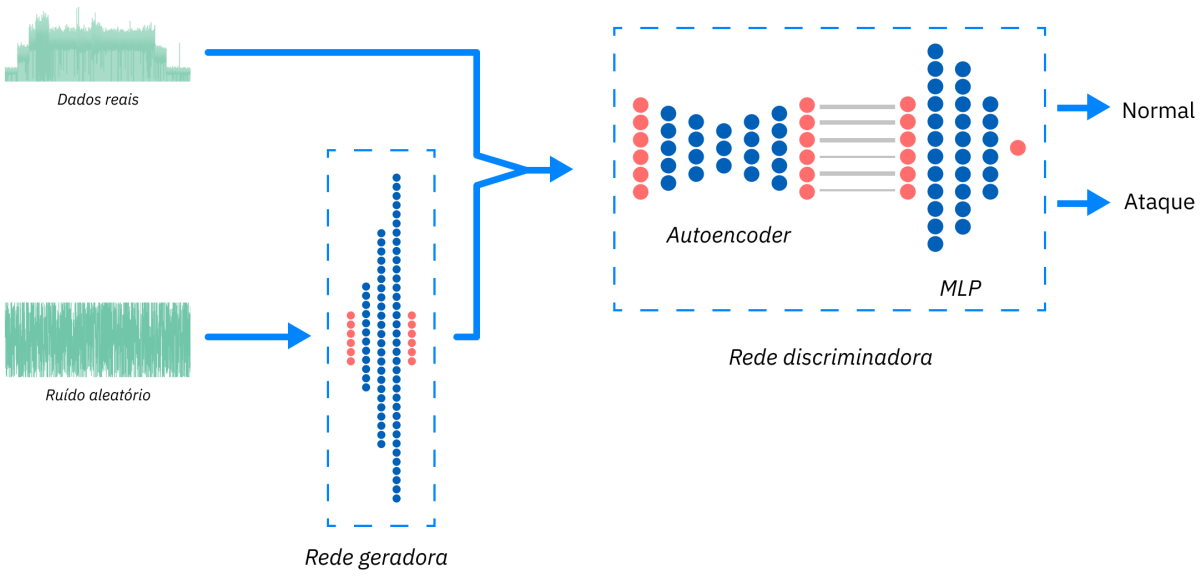


Figura 13 – Representação da *GAN* combinada com o *Autoencoder*

Estruturalmente, podemos observar na Figura 13 que da mesma forma que na rede *GAN* anterior, as redes *MLP* presentes no gerador e no discriminador continuam as mesmas que descritas nas Tabelas 12 e 11. Mas agora, com a adição da rede *Autoencoder* anteriormente desenvolvida, sendo inserida já treinada na rede discriminadora, recebendo as entradas antes de serem passadas para a *MLP* discriminadora final.

O objetivo é que essa inclusão facilite o trabalho final ao previamente filtrar as entradas e funcionar como um filtro amplificador, distorcendo ainda mais entradas que sejam anômalas para a tomada de decisão realizada pela *MLP*.

Destaca-se que assim como um filtro estático, tanto para que não haja perda de função ou distorção, a rede *Autoencoder* é excluída de qualquer treinamento enquanto na GAN. Dessarte, mantendo seus valores previamente já treinados e idealmente forçando uma estabilidade na rede, dificultando a perda de gradiente ao reforçar padrões.

6.4.1 Resultados

Equitativamente ao treinamento da GAN, também realizou-se uma *Grid Search* na busca de encontrar os melhores hiperparâmetros. Novamente, considerou-se a execução de até 4096 épocas com os tamanhos de *batch* 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192 e 16386 para cada cenário. Ao final da execução, o melhor resultado obtido foi com a execução de 18 épocas com *batchs* de 32.

Tabela 14 – Resultados obtidos com a *GAN-Autoencoder* em comparação com o *GAN*

	Conjunto de dados A				Conjunto de dados B			
	Dia 2		Dia 3		Dia 2		Dia 3	
Acurácia	99.91%	-0.06%	99.94%	-0.03%	96.48%	-3.39%	95.52%	-2.50%
Precisão	99.97%	-0.02%	100%	–	100%	–	97.90%	+0.02%
Revocação	99.93%	-0.04%	99.94%	-0.03%	96.53%	-3.33%	96.54%	-3.33%
F1	99.95%	-0.03%	99.97%	-0.02%	98.23%	-1.70%	97.22%	-1.65%

Os resultados são apresentados na Tabela 14. Trazendo números interessantes, pois embora não tenha acontecido uma melhora dos resultados, o cenário A trouxe mudanças desprezíveis e o cenário B, ainda que tenha sofrido uma maior defasagem, continua com números aceitáveis e bem maiores dos que originalmente encontrados nas execuções do *Autoencoder*.

6.5 Comparativo final

Em conclusão, podemos comparar os resultados obtidos pelos 3 modelos apresentados. Na Figura 14, onde são apresentadas as métricas obtidas ao longo do dia 2 no cenário A, é possível observar o destaque dos resultados da *GAN* em todos os atributos aferidos.

Na imagem, embora haja maiores diferenças com os valores obtidos pelo *Autoencoder*, todas as redes trazem desempenhos altos sem maiores problemas.

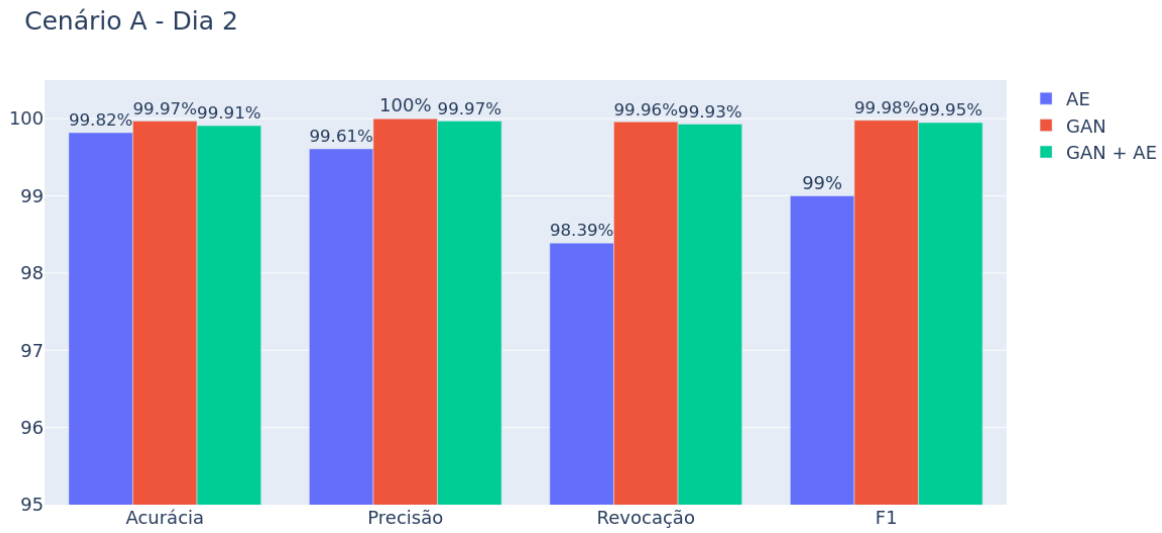


Figura 14 – Resultados obtidos no Cenário A - Dia 2

No gráfico apresentado na Figura 15 com os resultados do dia 3 no cenário A, os resultados se tornam mais parelhos e muito mais próximos. Destaca-se a diferença que ocorre entre as redes *Autoencoder* e as outras duas baseadas na *GAN*: enquanto uma tende para uma maior precisão e traz levemente uma menor revocação, a outra apresenta um comportamento inverso. Enquanto a rede *Autoencoder* acusou erroneamente alguns períodos de tráfego normal como ataque, as redes *GAN* e *GAN* com *Autoencoder* deixaram de acusar alguns dos ataques que ocorreram.

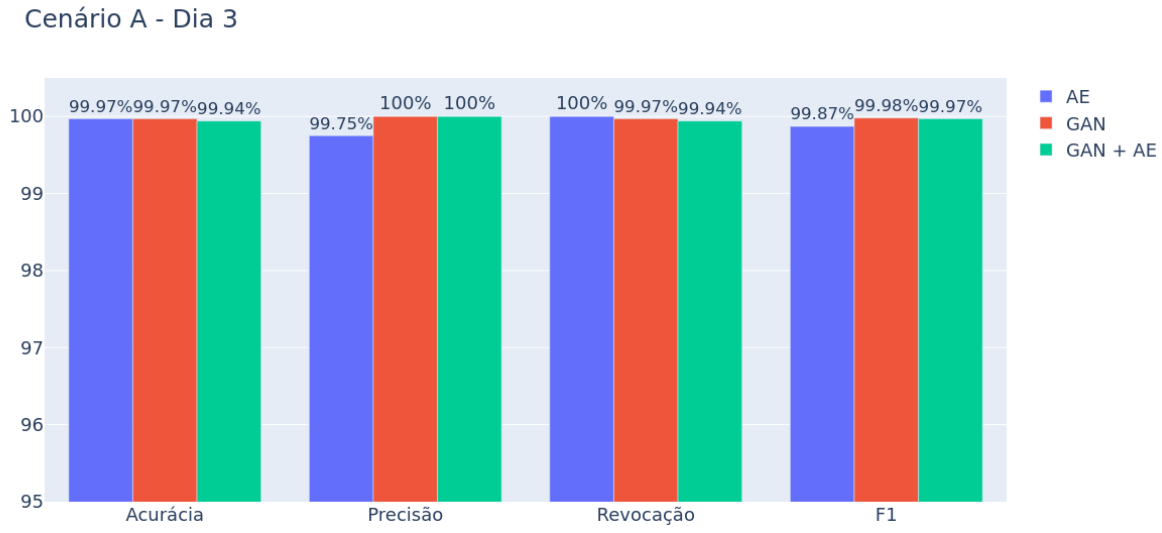


Figura 15 – Resultados obtidos no Cenário A - Dia 3

No segundo dia do cenário B, pontuado pela Figura 16, os resultados começam a mostrar maiores diferenças, ressaltando a dificuldade já conhecida da rede *Autoencoder* neste conjunto de dados. Contudo, diferentemente do padrão que se seguiu no cenário A, agora se torna visível uma superioridade da *GAN* em relação a *GAN* com o *Autoencoder* em todas as métricas, com exceção da precisão que trouxe um empate.

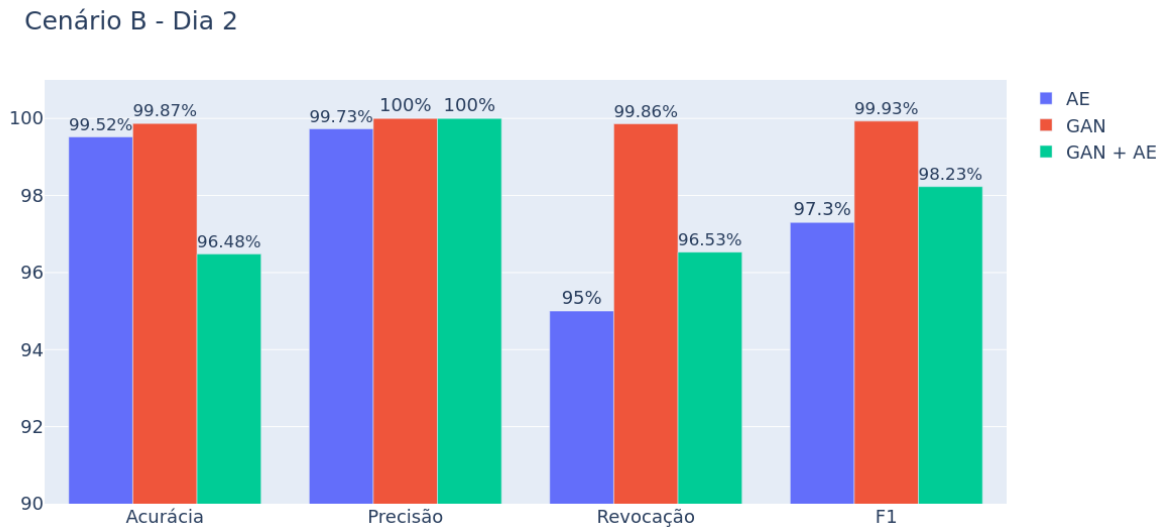


Figura 16 – Resultados obtidos no Cenário B - Dia 2

Por fim, no terceiro dia do cenário B na Figura 17, há uma acentuação máxima dos resultados. Além do padrão já apresentado no dia 2 com uma leve superioridade de resultados para a *GAN* contra a *GAN* com *Autoencoder*, é chamado a atenção para a revocação obtida pelo *Autoencoder*. Sendo ela, a menor de todas as métricas ao longo de todos os testes realizados, a revocação de 44.38% mostra uma completa falha ao acusar os ataques. Culminando assim, em uma métrica *F1* bem reduzida.

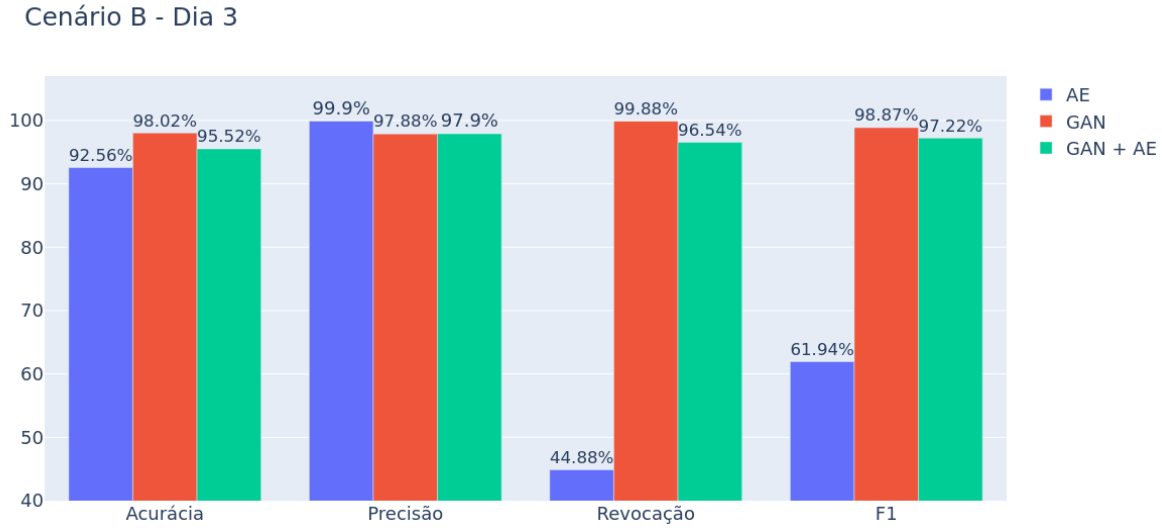


Figura 17 – Resultados obtidos no Cenário B - Dia 3

Resumidamente, no decorrer dos testes realizados, se torna perceptível uma tendência comum de uma melhor performance ser alcançada pela *GAN*, seguida da *GAN* com *Autoencoder* e por último, pelo próprio *Autoencoder*. Particularmente ainda, de forma mais explícita nos testes do Cenário B, criando um distanciação ainda maior no dia 3 com uma revocação, e consequentemente F1, extremamente baixa em relação aos outros patamares observados.

Em suma, também verifica-se como a hipótese anteriormente levantada na criação da *GAN* com *Autoencoder* não necessariamente se torna verdade. Dado que o modelo não atingiu patamares superiores e nem se provou ter uma performance superior com menos treinamento. Concluindo que, ao contrário do que se esperava, a inserção da rede *Autoencoder* já treinada de forma imutável na rede se mostrou como uma completa redefinição da arquitetura e funcionamento rede, diferentemente da ideia que seria uma mudança puramente positiva e adicional.

7 CONCLUSÃO

A área da cibersegurança, especialmente no ramo de detecção de ataques em redes, como pesquisado neste trabalho, é um tópico pesquisado a anos. Situação essa que com os avanços exigidos e feitos pela tecnologia, somente irá se intensificar.

Se trata de um cabo de guerra infinito onde quanto mais a pesquisa de defesa avançar, mais complexo e sofisticados serão os ataques. Neste trabalho, foi proposta a utilização de duas redes diferentes além de uma combinação delas. As redes *Autoencoder*, embora já utilizadas para detecção de anomalias, trazem a problemática da criação de um padrão rígido o suficiente para detectar ataques, mas flexível o suficiente para o dia a dia das redes de hoje. As redes *GAN* por outro lado, apesar de originalmente serem redes generativas com propósitos ao redor de áreas gráficas, vem se demonstrando uma das áreas mais promissoras dos últimos avanços na inteligência artificial. Fato este que traz a experimentação dessa rede para este trabalho.

Ao final dos testes realizados, é visível o potencial que as redes *GAN* assim como suas variações podem trazer, alcançando métricas extremante altas e consistentes em condições ideais. Contudo, não invalidando a eficácia do *Autoencoder* que, ainda com dificuldades em alguns dos cenários apresentados, poderia ter sido melhorado em diversos aspectos, tanto em sua rede neural como, em especial, em seu algoritmo de *threshold* para que se alcançasse melhores resultados. Evidência disso sendo a GAN combinada com o Autoencoder que, ao permitir um gerenciamento próprio de decisão dentro da rede, não refletiu as quedas abruptas de desempenho que o mesmo teve quando executado com o Chebyshev.

Em trabalhos futuros, é direção certa que a exploração das arquiteturas *GAN* deva trazer resultados ainda mais promissores. Para tal, é interessante que se realizem testes com implementações mais próximas do estado da arte, que hoje, se encontram quase que exclusivamente aplicadas a áreas gráficas. Neste trabalho, ao combinar-se as redes, foram utilizadas 2 *MLP* e um *Autoencoder* simples. Contudo, já existem técnicas e arquiteturas muito mais complexas que juntam redes como as *VAE* (*Variational Autoencoder*) com fluxos mais intrínsecos que se atrelam diretamente com transferências de pesos nas arquiteturas *GAN* [89]. Como a exemplo, as variantes *BiGAN* [90] e *AnoGAN* [63].

REFERÊNCIAS

[1] MAHDAVIFAR, S.; GHORBANI, A. A. Application of deep learning to cybersecurity: A survey. *Neurocomputing*, v. 347, p. 149–176, 2019. ISSN 0925-2312. Disponível em: <https://doi.org/10.1016/j.neucom.2019.02.056>.

[2] SCARANTI, G. F. et al. Artificial immune systems and fuzzy logic to detect flooding attacks in software-defined networks. *IEEE Access*, v. 8, p. 100172–100184, 2020. Disponível em: <https://doi.org/10.1109/ACCESS.2020.2997939>.

[3] NOVAES, M. P. et al. Long short-term memory and fuzzy logic for anomaly detection and mitigation in software-defined network environment. *IEEE Access*, v. 8, p. 83765–83781, 2020. Disponível em: <https://doi.org/10.1109/ACCESS.2020.2992044>.

[4] HAMAMOTO, A. H. et al. Network anomaly detection system using genetic algorithm and fuzzy logic. *Expert Systems with Applications*, v. 92, p. 390–402, 2018. ISSN 0957-4174. Disponível em: <https://doi.org/10.1016/j.eswa.2017.09.013>.

[5] ASSIS, M. V. D. et al. Fast defense system against attacks in software defined networks. *IEEE Access*, v. 6, p. 69620–69639, 2018. Disponível em: <https://doi.org/10.1109/ACCESS.2018.2878576>.

[6] CARVALHO, L. F. et al. A novel anomaly detection system to assist network management in sdn environment. In: *2017 IEEE International Conference on Communications (ICC)*. [s.n.], 2017. p. 1–6. Disponível em: <https://doi.org/10.1109/ICC.2017.7997214>.

[7] CARVALHO, L. F. et al. An ecosystem for anomaly detection and mitigation in software-defined networking. *Expert Systems with Applications*, v. 104, p. 121–133, 2018. ISSN 0957-4174. Disponível em: <https://doi.org/10.1016/j.eswa.2018.03.027>.

[8] ASSIS, M. V. O. D. et al. A game theoretical based system using holt-winters and genetic algorithm with fuzzy logic for dos/ddos mitigation on sdn networks. *IEEE Access*, v. 5, p. 9485–9496, 2017. Disponível em: <https://doi.org/10.1109/ACCESS.2017.2702341>.

[9] FADLULLAH, Z. M. et al. State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems. *IEEE Communications Surveys Tutorials*, v. 19, n. 4, p. 2432–2455, 2017. Disponível em: <https://doi.org/10.1109/COMST.2017.2707140>.

[10] VINAYAKUMAR, R. et al. Deep learning approach for intelligent intrusion detection system. *IEEE Access*, v. 7, p. 41525–41550, 2019. Disponível em: <https://doi.org/10.1109/ACCESS.2019.2895334>.

[11] da Costa, K. A. et al. Internet of things: A survey on machine learning-based intrusion detection approaches. *Computer Networks*, v. 151, p. 147–157, 2019. ISSN 1389-1286. Disponível em: <https://doi.org/10.1016/j.comnet.2019.01.023>.

[12] CARVALHO, L. F. et al. Unsupervised learning clustering and self-organized agents applied to help network management. *Expert Systems with Applications*, v. 54, p. 29–47, 2016. ISSN 0957-4174. Disponível em: <https://doi.org/10.1016/j.eswa.2016.01.032>.

[13] CHAABOUNI, N. et al. Network intrusion detection for iot security based on learning techniques. *IEEE Communications Surveys Tutorials*, v. 21, n. 3, p. 2671–2701, 2019. Disponível em: <https://doi.org/10.1109/COMST.2019.2896380>.

[14] NOVAES, M. P. et al. Adversarial deep learning approach detection and defense against ddos attacks in sdn environments. *Future Generation Computer Systems*, v. 125, p. 156–167, 2021. ISSN 0167-739X. Disponível em: <https://doi.org/10.1016/j.future.2021.06.047>.

[15] ASSIS, M. V. et al. A gru deep learning system against attacks in software defined networks. *Journal of Network and Computer Applications*, v. 177, p. 102942, 2021. ISSN 1084-8045. Disponível em: <https://doi.org/10.1016/j.jnca.2020.102942>.

[16] de Assis, M. V. et al. Near real-time security system applied to sdn environments in iot networks using convolutional neural network. *Computers Electrical Engineering*, v. 86, p. 106738, 2020. ISSN 0045-7906. Disponível em: <https://doi.org/10.1016/j.compeleceng.2020.106738>.

[17] ZERBINI, C. B. et al. Wavelet against random forest for anomaly mitigation in software-defined networking. *Applied Soft Computing*, v. 80, p. 138–153, 2019. ISSN 1568-4946. Disponível em: <https://doi.org/10.1016/j.asoc.2019.02.046>.

[18] PENA, E. H. et al. Anomaly detection using the correlational paraconsistent machine with digital signatures of network segment. *Information Sciences*, v. 420, p. 313–328, 2017. ISSN 0020-0255. Disponível em: <https://doi.org/10.1016/j.ins.2017.08.074>.

[19] SOMMER, R.; PAXSON, V. Outside the closed world: On using machine learning for network intrusion detection. In: *2010 IEEE Symposium on Security and Privacy*. [s.n.], 2010. p. 305–316. Disponível em: <https://doi.org/10.1109/SP.2010.25>.

[20] FARRIS, I. et al. A survey on emerging sdn and nfv security mechanisms for iot systems. *IEEE Communications Surveys Tutorials*, v. 21, n. 1, p. 812–837, 2019. Disponível em: <https://doi.org/10.1109/COMST.2018.2862350>.

[21] AHMED, M.; Naser Mahmood, A.; HU, J. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, v. 60, p. 19–31, 2016. ISSN 1084-8045. Disponível em: <https://doi.org/10.1016/j.jnca.2015.11.016>.

[22] BHUYAN, M. H.; BHATTACHARYYA, D. K.; KALITA, J. K. Network anomaly detection: Methods, systems and tools. *IEEE Communications Surveys Tutorials*, v. 16, n. 1, p. 303–336, 2014. Disponível em: <https://doi.org/10.1109/SURV.2013.052213.00046>.

[23] BARNETT, V.; LEWIS, T. *Outliers in statistical data.* 2nd edition. ed. [S.l.]: John Wiley & Sons Ltd., 1978.

[24] HAWKINS, D. M. *Identification of Outliers*. Springer Netherlands, 1980. Disponível em: <https://doi.org/10.1007/978-94-015-3994-4>.

[25] CHANDOLA, V.; BANERJEE, A.; KUMAR, V. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, ACM New York, NY, USA, v. 41, n. 3, p. 1–58, 2009. Disponível em: <https://doi.org/10.1145/1541880.1541882>.

[26] HOQUE, N. et al. Network attacks: Taxonomy, tools and systems. *Journal of Network and Computer Applications*, v. 40, p. 307–324, 2014. ISSN 1084-8045. Disponível em: <https://doi.org/10.1016/j.jnca.2013.08.001>.

[27] FERNANDES, G. et al. A comprehensive survey on network anomaly detection. *Telecommunication Systems*, Springer, v. 70, n. 3, p. 447–489, 2019. Disponível em: <https://doi.org/10.1007/s11235-018-0475-8>.

[28] BARFORD, P. et al. A signal analysis of network traffic anomalies. In: *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment*. New York, NY, USA: Association for Computing Machinery, 2002. (IMW '02), p. 71–82. ISBN 158113603X. Disponível em: <https://doi.org/10.1145/637201.637210>.

[29] SONG, X. et al. Conditional anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, v. 19, n. 5, p. 631–645, 2007. Disponível em: <https://doi.org/10.1109/TKDE.2007.1009>.

[30] MARNERIDES, A.; SCHAEFFER-FILHO, A.; MAUTHE, A. Traffic anomaly diagnosis in internet backbone networks: A survey. *Computer Networks*, v. 73, p. 224–243, 2014. ISSN 1389-1286. Disponível em: <https://doi.org/10.1016/j.comnet.2014.08.007>.

[31] LöF, A.; NELSON, R. Annotating network trace data for anomaly detection research. In: *39th Annual IEEE Conference on Local Computer Networks Workshops*. [s.n.], 2014. p. 679–684. Disponível em: <https://doi.org/10.1109/LCNW.2014.6927720>.

[32] GHORBANI, A. A.; LU, W.; TAVALLAEE, M. Network attacks. In: _____. *Network Intrusion Detection and Prevention: Concepts and Techniques*. Boston, MA: Springer US, 2010. p. 1–25. ISBN 978-0-387-88771-5. Disponível em: <https://doi.org/10.1007/978-0-387-88771-5_1>.

[33] HANSMAN, S.; HUNT, R. A taxonomy of network and computer attacks. *Computers Security*, v. 24, n. 1, p. 31–43, 2005. ISSN 0167-4048. Disponível em: <https://doi.org/10.1016/j.cose.2004.06.011>.

[34] ELSAYED, M. S. et al. Ddosnet: A deep-learning model for detecting network attacks. In: *2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*. [s.n.], 2020. p. 391–396. Disponível em: <https://doi.org/10.1109/WoWMoM49955.2020.00072>.

[35] ZARGAR, S. T.; JOSHI, J.; TIPPER, D. A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks. *IEEE Communications Surveys Tutorials*, v. 15, n. 4, p. 2046–2069, 2013. Disponível em: <https://doi.org/10.1109/SURV.2013.031413.00127>.

[36] KOLIAS, C. et al. Ddos in the iot: Mirai and other botnets. *Computer*, v. 50, n. 7, p. 80–84, 2017. Disponível em: <https://doi.org/10.1109/MC.2017.201>.

[37] MOHAMMED, S. S. et al. A new machine learning-based collaborative ddos mitigation mechanism in software-defined network. In: *2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. [s.n.], 2018. p. 1–8. Disponível em: <https://doi.org/10.1109/WiMOB.2018.8589104>.

[38] LEE, C. B.; ROEDEL, C.; SILENOK, E. Detection and characterization of port scan attacks. *Univeristy of California, Department of Computer Science and Engineering*, 2003.

[39] JUNG, J. et al. Fast portscan detection using sequential hypothesis testing. In: *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004.* [s.n.], 2004. p. 211–225. Disponível em: <https://doi.org/10.1109/SECPRI.2004.1301325>.

[40] XU, Y. et al. Machine learning in construction: From shallow to deep learning. *Developments in the Built Environment*, v. 6, p. 100045, 2021. ISSN 2666-1659. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2666165921000041>.

[41] MALIK, P. et al. Overview of artificial intelligence in medicine. *Journal of family medicine and primary care*, Wolters Kluwer–Medknow Publications, v. 8, n. 7, p. 2328, 2019. Disponível em: <https://doi.org/10.4103/jfmpc.jfmpc_440_19>.

[42] AKHTAR, M.; MORIDPOUR, S. A review of traffic congestion prediction using artificial intelligence. *Journal of Advanced Transportation*, Hindawi, v. 2021, 2021. Disponível em: <https://doi.org/10.1155/2021/8878011>.

[43] WANG, C.-X. et al. Artificial intelligence enabled wireless networking for 5g and beyond: Recent advances and future challenges. *IEEE Wireless Communications*, IEEE, v. 27, n. 1, p. 16–23, 2020. Disponível em: <https://doi.org/10.1109/MWC.001.1900292>.

[44] MCCARTHY, J. What is artificial intelligence. *Project JMC - Stanford University*, 2007. Disponível em: <http://jmc.stanford.edu/articles/whatisai/whatisai.pdf>.

[45] AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: What's the Difference? <https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks>. Accessed: 2022-02-28.

[46] NAQA, I. E.; MURPHY, M. J. What is machine learning? In: _____. *Machine Learning in Radiation Oncology: Theory and Applications*. Cham: Springer International Publishing, 2015. p. 3–11. ISBN 978-3-319-18305-3. Disponível em: <https://doi.org/10.1007/978-3-319-18305-3_1>.

[47] MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. *Foundations of machine learning*. [S.l.]: MIT press, 2018.

[48] YAO, X.; LIU, Y. Machine learning. In: _____. *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Boston, MA: Springer US, 2014. p. 477–517. ISBN 978-1-4614-6940-7. Disponível em: <https://doi.org/10.1007/978-1-4614-6940-7_17>.

[49] ANGRA, S.; AHUJA, S. Machine learning and its applications: A review. In: *2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC)*. [s.n.], 2017. p. 57–60. Disponível em: <https://doi.org/10.1109/ICBDACI.2017.8070809>.

[50] MALAIYA, R. K. et al. An empirical evaluation of deep learning for network anomaly detection. In: *2018 International Conference on Computing, Networking and Communications (ICNC)*. [s.n.], 2018. p. 893–898. Disponível em: <https://doi.org/10.1109/ICCNC.2018.8390278>.

[51] WU, S. X.; BANZHAF, W. The use of computational intelligence in intrusion detection systems: A review. *Applied Soft Computing*, v. 10, n. 1, p. 1–35, 2010. ISSN 1568-4946. Disponível em: <https://doi.org/10.1016/j.asoc.2009.06.019>.

[52] PANG, G. et al. Deep learning for anomaly detection: A review. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 54, n. 2, mar. 2021. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3439950>.

[53] UTGOFF, P. E.; STRACUZZI, D. J. Many-layered learning. *Neural computation*, MIT Press One Rogers Street, v. 14, n. 10, p. 2497–2529, 2002. Disponível em: <https://doi.org/10.1162/08997660260293319>.

[54] ZENG, Y. et al. Deep-full-range: A deep learning based network encrypted traffic classification and intrusion detection framework. *IEEE Access*, PP, p. 1–1, 01 2019. Disponível em: <https://doi.org/10.1109/ACCESS.2019.2908225>.

[55] YUAN, X. et al. Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, v. 30, n. 9, p. 2805–2824, 2019. Disponível em: <https://doi.org/10.1109/TNNLS.2018.2886017>.

[56] FERRAG, M. A. et al. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, v. 50, p. 102419, 2020. ISSN 2214-2126. Disponível em: <https://doi.org/10.1016/j.jisa.2019.102419>.

[57] BENGIO, Y. *Learning deep architectures for AI*. [S.l.]: Now Publishers Inc, 2009.

[58] GOODFELLOW, I. et al. Generative adversarial nets. *Advances in neural information processing systems*, v. 27, 2014. Disponível em: <https://doi.org/10.48550/arXiv.1406.2661>.

[59] AGGARWAL, A.; MITTAL, M.; BATTINENI, G. Generative adversarial network: An overview of theory and applications. *International Journal of Information Management Data Insights*, v. 1, n. 1, p. 100004, 2021. ISSN 2667-0968. Disponível em: <https://doi.org/10.1016/j.jjimei.2020.100004>.

[60] CRESWELL, A.; BHARATH, A. A. Inverting the generator of a generative adversarial network. *IEEE Transactions on Neural Networks and Learning Systems*, v. 30, n. 7, p. 1967–1974, 2019. Disponível em: <https://doi.org/10.1109/TNNLS.2018.2875194>.

[61] MATTIA, F. D. et al. *A Survey on GANs for Anomaly Detection*. arXiv, 2019. Disponível em: <https://doi.org/10.48550/ARXIV.1906.11632>.

[62] AKCAY, S.; ATAPOUR-ABARGHOUEI, A.; BRECKON, T. P. *GANomaly: Semi-Supervised Anomaly Detection via Adversarial Training*. arXiv, 2018. Disponível em: <https://doi.org/10.48550/ARXIV.1805.06725>.

[63] SCHLEGL, T. et al. *Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery*. arXiv, 2017. Disponível em: <https://doi.org/10.48550/ARXIV.1703.05921>.

[64] ZENATI, H. et al. *Efficient GAN-Based Anomaly Detection*. arXiv, 2018. Disponível em: <https://doi.org/10.48550/ARXIV.1802.06222>.

[65] HUANG, X. et al. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review*, v. 37, p. 100270, 2020. ISSN 1574-0137. Disponível em: <https://doi.org/10.1016/j.cosrev.2020.100270>.

[66] PAWLICKI, M.; CHORAś, M.; KOZIK, R. Defending network intrusion detection systems against adversarial evasion attacks. *Future Generation Computer Systems*, v. 110, p. 148–154, 2020. ISSN 0167-739X. Disponível em: <https://doi.org/10.1016/j.future.2020.04.013>.

[67] RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning internal representations by error propagation. In: _____. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986. p. 318–362. ISBN 026268053X.

[68] ZHOU, C.; PAFFENROTH, R. C. Anomaly detection with robust deep autoencoders. In: *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. [s.n.], 2017. p. 665–674. Disponível em: <https://doi.org/10.1145/3097983.3098052>.

[69] CHEN, Z. et al. Autoencoder-based network anomaly detection. In: *2018 Wireless Telecommunications Symposium (WTS)*. [s.n.], 2018. p. 1–5. Disponível em: <https://doi.org/10.1109/WTS.2018.8363930>.

[70] BANK, D.; KOENIGSTEIN, N.; GIRYES, R. *Autoencoders*. 2021. Disponível em: <https://doi.org/10.48550/arXiv.2003.05991>.

[71] PU, Y. et al. Variational autoencoder for deep learning of images, labels and captions. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2016. (NIPS'16), p. 2360–2368. ISBN 9781510838819. Disponível em: <https://doi.org/10.48550/arXiv.1609.08976>.

[72] SAKURADA, M.; YAIRI, T. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In: *Proceedings of the MLSDA 2014 2nd workshop on machine learning for sensory data analysis*. [s.n.], 2014. p. 4–11. Disponível em: <https://doi.org/10.1145/2689746.2689747>.

[73] KHAN, F. A. et al. A novel two-stage deep learning model for efficient network intrusion detection. *IEEE Access*, v. 7, p. 30373–30385, 2019. Disponível em: <https://doi.org/10.1109/ACCESS.2019.2899721>.

[74] LEE, W.; STOLFO, S. J. Data mining approaches for intrusion detection. In: *Proceedings of the 7th Conference on USENIX Security Symposium - Volume 7*. USA: USENIX Association, 1998. (SSYM'98), p. 6.

[75] BOSTANI, H.; SHEIKHAN, M. Hybrid of anomaly-based and specification-based ids for internet of things using unsupervised opf based on mapreduce approach. *Computer Communications*, v. 98, p. 52–71, 2017. ISSN 0140-3664. Disponível em: <https://doi.org/10.1016/j.comcom.2016.12.001>.

[76] HODO, E. et al. *Shallow and Deep Networks Intrusion Detection System: A Taxonomy and Survey*. 2017. Disponível em: <https://doi.org/10.48550/arXiv.1701.02145>.

[77] LIAO, H.-J. et al. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, v. 36, n. 1, p. 16–24, 2013. ISSN 1084-8045. Disponível em: <https://doi.org/10.1016/j.jnca.2012.09.004>.

[78] DEBAR, H.; DACIER, M.; WESPI, A. Towards a taxonomy of intrusion-detection systems. *Computer Networks*, v. 31, n. 8, p. 805–822, 1999. ISSN 1389-1286. Disponível em: <https://doi.org/10.1016/S1389-1286(98)00017-6>.

[79] JYOTHSNA, V.; PRASAD, R.; PRASAD, K. M. A review of anomaly based intrusion detection systems. *International Journal of Computer Applications*, Citeseer, v. 28, n. 7, p. 26–35, 2011. Disponível em: <http://dx.doi.org/10.5120/3399-4730>.

[80] PROENçA, M. L. et al. The hurst parameter for digital signature of network segment. In: SOUZA, J. N. de; DINI, P.; LORENZ, P. (Ed.). *Telecommunications and Networking - ICT 2004*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 772–781. ISBN 978-3-540-27824-5. Disponível em: <http://dx.doi.org/10.1007/978-3-540-27824-5_103>.

[81] NIYAZ, Q.; SUN, W.; JAVAID, A. Y. A deep learning based DDoS detection system in software-defined networking (SDN). *ICST Transactions on Security and Safety*, European Alliance for Innovation n.o., v. 4, n. 12, p. 153515, dec 2017. Disponível em: <https://doi.org/10.4108/eai.28-12-2017.153515>.

[82] ANDRESINI, G.; APPICE, A.; MALERBA, D. Autoencoder-based deep metric learning for network intrusion detection. *Information Sciences*, v. 569, p. 706–727, 2021. ISSN 0020-0255. Disponível em: <https://doi.org/10.1016/j.ins.2021.05.016>.

[83] NIU, Z.; YU, K.; WU, X. Lstm-based vae-gan for time-series anomaly detection. *Sensors*, v. 20, n. 13, 2020. ISSN 1424-8220. Disponível em: <https://doi.org/10.3390/s20133738>.

[84] ZENATI, H. et al. Adversarially learned anomaly detection. In: *2018 IEEE International Conference on Data Mining (ICDM)*. [s.n.], 2018. p. 727–736. Disponível em: <https://doi.org/10.1109/ICDM.2018.00088>.

[85] SHANNON, C. E. A mathematical theory of communication. *The Bell system technical journal*, Nokia Bell Labs, v. 27, n. 3, p. 379–423, 1948.

[86] PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Disponível em: <https://doi.org/10.5555/1953048.2078195>.

[87] AMIDAN, B.; FERRYMAN, T.; COOLEY, S. Data outlier detection using the chebyshev theorem. In: *2005 IEEE Aerospace Conference*. [s.n.], 2005. p. 3814–3819. Disponível em: <https://doi.org/10.1109/AERO.2005.1559688>.

[88] TAYLOR, C.; ALVES-FOSS, J. An empirical analysis of nate - network analysis of anomalous traffic events. *Proc. New Security Paradigms Workshop*, 09 2002. Disponível em: <https://doi.org/10.1145/844102.844106>.

[89] LUO, Y.; WANG, X.; POURPANAH, F. Dual vaegan: A generative model for generalized zero-shot learning. *Applied Soft Computing*, v. 107, p. 107352, 2021. ISSN 1568-4946. Disponível em: <https://doi.org/10.1016/j.asoc.2021.107352>.

[90] DONAHUE, J.; KRäHENBüHL, P.; DARRELL, T. *Adversarial Feature Learning*. arXiv, 2016. Disponível em: <https://doi.org/10.48550/ARXIV.1605.09782>.