Universidade Estadual de Londrina

VITOR GABRIEL DA SILVA RUFFO

DETECÇÃO DE ANOMALIAS EM REDES DE COMPUTADORES UTILIZANDO REDES ADVERSÁRIAS GENERATIVAS

LONDRINA
2023

VITOR GABRIEL DA SILVA RUFFO

DETECÇÃO DE ANOMALIAS EM REDES DE COMPUTADORES UTILIZANDO REDES ADVERSÁRIAS GENERATIVAS

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação do Departamento de Computação da Universidade Estadual de Londrina, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Mario Lemes Proença Jr.

LONDRINA
2023

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

R925d Ruffo, Vitor Gabriel da Silva.
Detecção de anomalias em redes de computadores utilizando redes adversárias generativas / Vitor Gabriel da Silva Ruffo. - Londrina, 2023.
100 f. : il.

Orientador: Mario Lemes Proença Jr..
Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Universidade Estadual de Londrina, Centro de Ciências Exatas, Graduação em Ciência da Computação, 2023.
Inclui bibliografia.

1. Segurança de redes - TCC. 2. Detecção de anomalias - TCC. 3. Detecção de intrusões - TCC. 4. Rede Adversária Generativa - TCC. I. Lemes Proença Jr., Mario. II. Universidade Estadual de Londrina. Centro de Ciências Exatas. Graduação em Ciência da Computação. III. Título.

CDU 519

VITOR GABRIEL DA SILVA RUFFO

DETECÇÃO DE ANOMALIAS EM REDES DE COMPUTADORES UTILIZANDO REDES ADVERSÁRIAS GENERATIVAS

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação do Departamento de Computação da Universidade Estadual de Londrina, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA

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

Prof. Dr. Elieser Botelho Manhas Jr.
Universidade Estudual de Londrina

Ms. Daniel Matheus Brandão Lent
Universidade Estudual de Londrina

Londrina, 30 de maio de 2023.

Este trabalho é dedicado aos meus pais e avós que sempre fizeram e continuam fazendo tudo o que está ao seu alcance para que eu me torne uma pessoa melhor.

AGRADECIMENTOS

Aos meus pais e avós, por todo o seu amor, esforço e anos de trabalho duro, possibilitando que eu estivesse aqui hoje escrevendo este texto.

Ao professor Dr. Mario Lemes Proença Jr., por toda a ajuda e orientação que me permitiu crescer imensamente como pessoa e profissional ao longo do desenvolvimento deste e de outros trabalhos.

Ao meu colega Ms. Daniel Matheus Brandão Lent, que trilha um caminho semelhante ao meu e que não mediu esforços para me ajudar a desenvolver os meus conhecimentos.

Aos meus colegas de curso, cujos esforços direcionados aos seus objetivos individuais sempre me motivaram a me dedicar cada vez mais nas minhas próprias metas.

Aos membros do grupo de pesquisa ORION da Universidade Estadual de Londrina, por todos os ensinamentos e por me inspirar a ser um pesquisador científico.

Aos professores do curso de Ciência da Computação, por me ensinar e proporcionar um amadurecimento imensurável no decorrer desses quatro anos de graduação.

Aos servidores da UEL, pelo seu valioso trabalho que permite que toda a universidade continue funcionando.

“O que sabemos é uma gota; o que ignoramos é um oceano“

Isaac Newton

RUFFO, V. G. S.. **Detecção de Anomalias em Redes de Computadores Utilizando Redes Adversárias Generativas**. 2023. 100f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2023.

RESUMO

Ao longo dos últimos anos, aplicações emergentes têm exigido serviços de rede cada vez mais complexos para funcionarem corretamente. Redes de computadores são constantemente expandidas e adaptadas para suprir as necessidades dessas aplicações. As redes também necessitam de mecanismos de segurança contra agentes maliciosos. Esses mecanismos objetivam a maximização da confidencialidade, integridade e disponibilidade dos serviços oferecidos pela rede. O investimento na segurança de rede evita falhas de funcionamento nas aplicações que dependem dos seus serviços e, consequentemente, perdas de lucro. Este trabalho de conclusão de curso tem como objetivo o estudo da aplicabilidade do modelo de Rede Adversária Generativa na implementação de um sistema de segurança para a detecção de anomalias e intrusões de redes. O sistema desenvolvido analisa o tráfego de rede segundo a segundo e alerta os administradores quando traços anômalos são identificados nos dados. Uma segunda versão desse sistema baseada em Rede Neural Recorrente *Gated Recurrent Unit* foi implementada para fins de comparação. Os desempenhos em detecção de anomalias dos sistemas foram calculados e comparados entre si utilizando métricas quantitativas a partir de uma base de dados comum. Os resultados indicam que ambos os sistemas podem identificar aproximadamente 99% dos ataques. O primeiro obtém as pontuações 0,9978 e 0,9977 para as métricas *F1-score* e *Matthews Correlation Coefficient*, respectivamente. O segundo pontua menos, atingindo os valores de 0,9947 e 0,9944. Essa diferença sugere que o sistema baseado no modelo generativo obtém um melhor balanço entre falsos positivos e falsos negativos do que o seu concorrente para o cenário de execução. A partir dos resultados, conclui-se que o modelo de Rede Adversária Generativa é uma alternativa viável na implementação de um sistema de detecção de intrusão de redes.

Palavras-chave: Segurança de redes. Detecção de anomalias. Detecção de intrusões. Rede Adversária Generativa.

RUFFO, V. G. S.. **Network Anomaly Detection based on Generative Adversarial Networks**. 2023. 100p. Final Project (Bachelor of Science in Computer Science) – State University of Londrina, Londrina, 2023.

ABSTRACT

In the past few years, emerging applications have been requiring increasingly complex network services to work correctly. Computer networks are constantly being expanded and adapted to supply those applications' needs. Networks also require security mechanisms to protect them against malicious agents. Those mechanisms' purpose is to maximize the confidentiality, integrity, and availability of the network services. Network security investment avoids disruptions on applications that rely on its services and, consequently, profit losses. This final project aims to study the applicability of the Generative Adversarial Network model in implementing a network intrusion detection system. The developed system analyzes network traffic every second and alerts the administrators when it identifies anomalous traits in data. A secondary security system based on the Gated Recurrent Unit Recurrent Neural Network was implemented for comparison purposes. Both systems' anomaly detection capabilities were calculated and compared against each other by using quantitative metrics and a common dataset. The results show that the two can identify nearly 99% of the attacks. The former gets 0,9978 and 0,9977 points on F1-score and Matthews Correlation Coefficient metrics, respectively. The second system scores less, securing 0,9947 and 0,9944 points. That difference suggests that the anomaly detection system based on the generative model has a better balance between false positives and false negatives for the execution scenario than its competitor. In conclusion, the Generative Adversarial Network model is a feasible alternative for implementing a network intrusion detection system.

Keywords: Network security. Anomaly detection. Intrusion detection. Generative Adversarial Network.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Adam	*Adaptive moment estimation*
AE	*Auto-Encoder*
AI	*Artificial Intelligence*
API	*Application Programming Interface*
AUROC	*Area Under the Receiver Operating Characteristic curve*
CNN	*Convolutional Neural Network*
DDoS	*Distribuited Denial of Service*
DL	*Deep Learning*
DNN	*Deep Neural Network*
FPR	*False Positive Rate*
GAN	*Generative Adversarial Network*
GRU	*Gated Recurrent Unit*
IoT	*Internet of Things*
LSTM	*Long Short-Term Memory*
MCC	*Matthews Correlation Coefficient*
ML	*Machine Learning*
NIDS	*Network Intrusion Detection System*
NN	*Neural Network*
ReLu	*Rectified Linear unit*
RNN	*Recurrent Neural Network*
ROC	*Receiver Operating Characteristic*
SDN	*Software Defined Network*
SGD	*Stochastic Gradient Descent*
TCN	*Temporal Convolutinal Network*

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos muitas tarefas realizadas pelo ser humano são facilitadas ou automatizadas com o uso de redes de computadores. Por exemplo, transmissão de vídeos através da Internet, digitalização de contas bancárias, comunicação interpessoal, trabalho remoto, veículos autônomos, casas inteligentes e transferências monetárias de maneira digital (Pix) [1], [2], [3], [4], [5].

As redes têm crescido e se tornado cada vez mais complexas para atender a demanda dos serviços prestados aos seus usuários. A expansão das redes dificulta o seu gerenciamento e manutenção devido à abundância de equipamentos heterogêneos que as compõem (*switches*, roteadores e *middle-boxes*) [6], [7]. Componentes de diferentes fabricantes possuem interfaces de programação distintas. Como resultado, atualizações nas políticas de rede e lógica de controle precisam ser definidas em cada um dos equipamentos de maneira individual. Essas mudanças são implementadas utilizando comandos de baixo nível específicos de cada dispositivo [8]. Deste modo, a arquitetura tradicional de rede se mostra inflexível e exige um alto custo humano e financeiro para escalar e se adaptar.

Com o propósito de solucionar essas limitações, foi desenvolvida uma arquitetura de rede capaz de atender aos requisitos das aplicações atuais e futuras, a chamada Rede Definida por Software (SDN, do inglês *Software Defined Network*) [9]. Para atingir esse objetivo, a rede SDN separa o plano de controle do plano de dados. Toda a lógica de controle da rede é transferida para um único ponto, o controlador [10]. Os sistemas de hardware que formam o plano de dados (*switches* e roteadores) se tornam responsáveis apenas pelo encaminhamento de pacotes. O controlador é programado utilizando uma única linguagem de alto nível para implementar as políticas da rede e lógica de controle. Ele torna-se responsável por transmiti-las para cada um dos equipamentos do plano de dados com base em um protocolo comum (*Openflow*) [11]. Assim, atualizações nas regras da rede são definidas uma única vez no controlador, que se encarrega de propagá-las aos demais nós. Isso facilita a gerência e a manutenção da rede, fazendo com que ela seja escalável e adaptativa.

Outro aspecto considerado pela comunidade científica para obter a qualidade nos serviços de rede é a segurança. Por estar tão presente no dia a dia das pessoas deste século, esses serviços possuem um alto valor e a sua falha pode causar enormes prejuízos [12], [13]. Geralmente, essas falhas são o produto de divergências estocásticas entre o comportamento de tráfego observado na rede e um *baseline*. Define-se como *baseline* o perfil de comportamento normal esperado do tráfego de rede. Esse perfil é previamente calculado com base no monitoramento de dados históricos [7], [14]. O tráfego que diverge do *baseline* recebe o nome de anomalia de rede [15]. Nem toda anomalia é causada por

um agente malicioso, como as decorrentes de erro de hardware, de software ou humano. Porém, é muito comum a ocorrência de ataques de redes, os quais são atividades anômalas causadas por usuários mal-intencionados. Essas ações maliciosas objetivam comprometer a confidencialidade, integridade ou disponibilidade da rede [16], [17]. Por exemplo, recentemente um ataque comprometeu um sistema importante de redes de computadores da Albânia. Isso obrigou as autoridades do governo a desligá-lo, impedindo o fornecimento de serviços públicos online para os moradores do país [18]. Assim, é de extrema importância a construção de meios para auxiliar na proteção das redes de computadores.

Uma abordagem de solução aceita e estudada na comunidade científica para promover a segurança contra ataques de redes são os Sistemas de Detecção de Intrusão de Redes (NIDS, do inglês *Network Intrusion Detection System*) [19], [20], [21], [22]. Esses sistemas agem monitorando o tráfego e gerando um aviso para o gerente de rede quando vestígios de comportamento anômalo são identificados. Seu principal intuito é o oposto dos agentes maliciosos: manter a confidencialidade, integridade e disponibilidade dos serviços de rede [15].

Entre os métodos apresentados na literatura para implementação de um NIDS, Aprendizado de Máquina tem se destacado. Ele obtém acurácia superior na resolução de problemas em comparação com outros modelos matemáticos [23], [24]. Exemplos desse método incluem K-Vizinhos Mais Próximos, Árvore de Decisão, e Rede Neural [25]. Dentro dessa área, as abordagens que mais têm ganhado atenção são as de Aprendizado Profundo por, geralmente, possuírem um melhor desempenho em relação às demais [26], [27]. Esses métodos são tipos especiais de Rede Neural, como por exemplo *Long Short-Term Memory (LSTM)* [28], *Convolutional Neural Network (CNN)* [29], *Auto-Encoder (AE)* e *Gated Recurrent Unit (GRU)* [30]. Uma técnica de Aprendizado Profundo promissora concebida no ano de 2014 é a Rede Adversária Generativa (GAN) [31], [32].

As redes estão em constante evolução, assim como variações de ataques conhecidos e novos ataques são continuamente concebidos por agentes maliciosos [33]. Deste modo, a área de detecção de anomalias de redes tem sido estudada há muitos anos pela comunidade científica [34], [35], [36]. Pesquisas recentes destacam problemas nesse campo que ainda não possuem uma solução definitiva [37], [38], [15], [33], [25], [39], [40]. Diante disso, este trabalho propõe-se a estudar a aplicação de GAN na implementação de um NIDS para a proteção de Redes Definidas por Software.

O restante deste trabalho está organizado da seguinte forma: são apresentados no capítulo dois os conceitos, métodos e técnicas necessários para o desenvolvimento do estudo proposto. Além disso, destaca-se no capítulo três a teoria de Rede Adversária Generativa. Em seguida, apresenta-se no capítulo quatro o estado da arte por meio de uma revisão de trabalhos similares publicados recentemente. Um estudo de caso é descrito no capítulo cinco. Por fim, as conclusões são expostas no capítulo seis.

2 FUNDAMENTAÇÃO TEÓRICO-METODOLÓGICA

2.1 Redes Definidas por Software

Em redes de computadores utiliza-se três planos de funcionalidade que trabalham em conjunto [41], como apresentado na Figura 1. O primeiro, o plano de dados, é responsável por realizar o encaminhamento de pacotes com base em regras pré-definidas. Essas regras representam as políticas e comportamentos esperados da rede, sendo especificadas no chamado plano de controle. O último plano é o de aplicação, onde os gerentes de rede atuam utilizando softwares para definir as regras do plano de controle e gerenciar a rede.

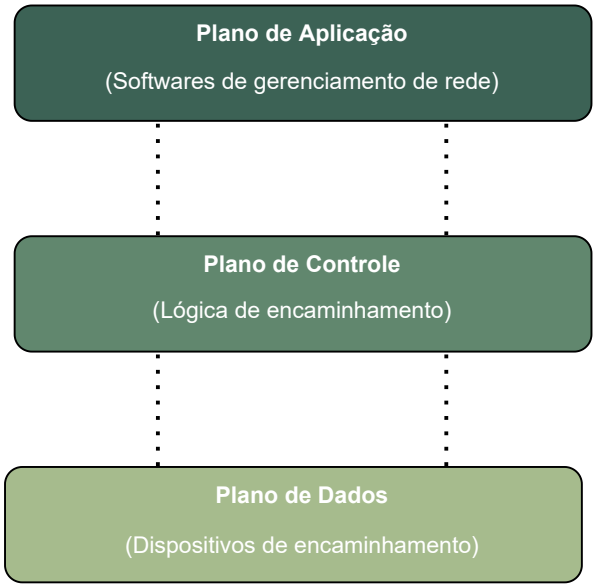

Figura 1 – Planos de funcionalidade de rede.

A arquitetura de redes de computadores tradicional implementa a abstração descrita anteriormente de uma maneira complexa e inflexível a mudanças, exigindo um alto custo de gerenciamento [42]. Neste paradigma o plano de controle é acoplado ao plano de dados. Assim, os equipamentos que formam a infraestrutura da rede (*switches* e roteadores) possuem o próprio mecanismo de controle e encaminhamento. A partir do plano de aplicação, os gerentes de redes configuram e gerenciam cada *switch* e roteador de maneira individual. Para isso, eles utilizam a interface de programação (API, do inglês *Application Programming Interface*) específica de cada vendedor de modo a expressar as políticas de rede desejadas [43]. A variedade de equipamentos aumenta o custo de manutenção e administração da rede por requerer conhecimento individualizado de cada uma de suas APIs. Como consequência, dificulta-se a evolução e adaptação da rede para poder atender a demandas cada vez mais exigentes das aplicações atuais [44].

As Redes Definidas por Software (SDN, do inglês *Software Defined Networks*) introduzem outra maneira de organizar a rede, como exemplificado na Figura 2. O plano de controle é separado do plano de dados e deslocado para um ponto logicamente centralizado: o controlador [45]. Assim, os equipamentos que formam a infraestrutura da rede passam a exercer apenas a função de encaminhamento de dados. O controlador se torna responsável por instalar a lógica de controle nesses encaminhadores utilizando uma API aberta (*OpenFlow*), a chamada interface *southbound* [46].

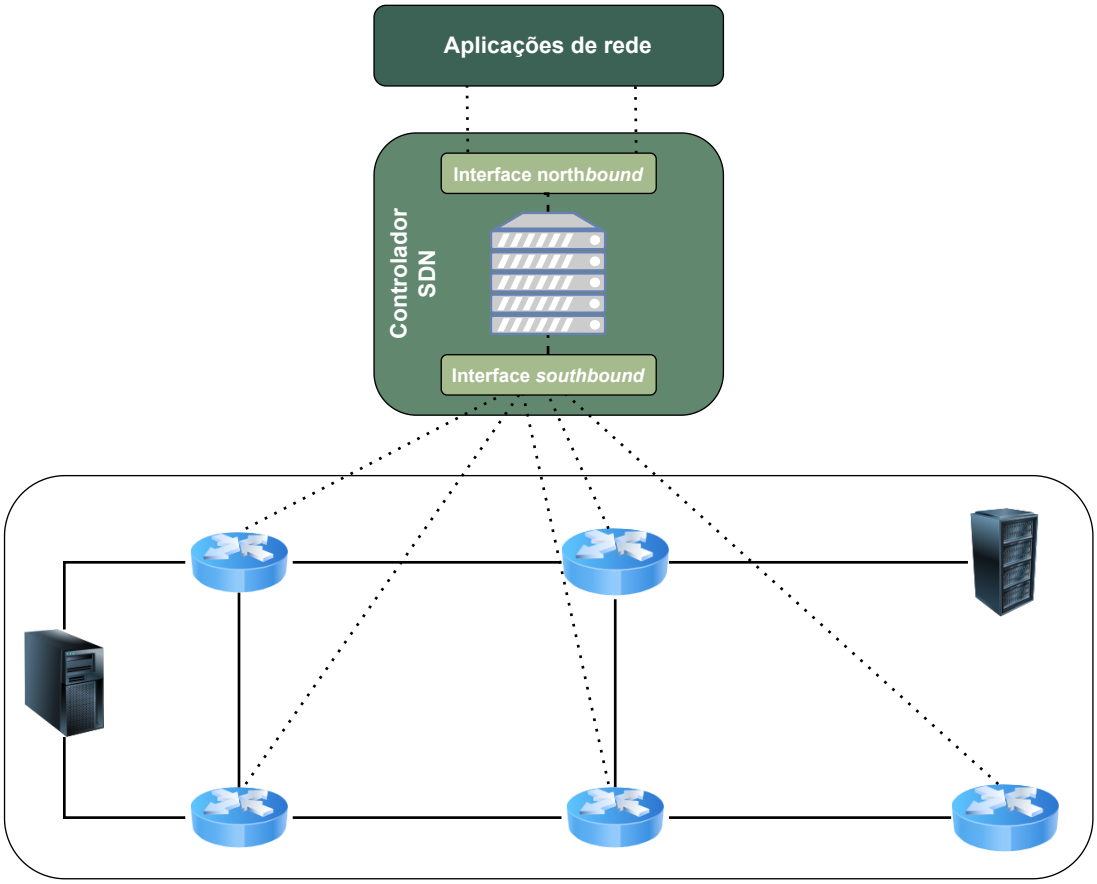

Figura 2 – Arquitetura SDN.

Os dispositivos do plano de dados encaminham os pacotes agrupados em fluxos IP, que representam uma troca de pacotes entre uma origem e um destino. Dentro desses equipamentos existem regras específicas para tratar pacotes de cada fluxo [43]. Além disso, eles também armazenam estatísticas dos fluxos que trafegaram pela rede. Essas informações podem ser posteriormente coletadas pelo controlador para fins de análise de tráfego [41].

A centralização de controle tem um papel parecido com o de um sistema operacional em um sistema computacional. O controlador fornece uma abstração do plano de dados por meio de uma API chamada de interface *northbound* [44]. Isso possibilita que os gerentes de rede programem o plano de dados a partir de uma única interface. Os

administradores conseguem expressar as políticas de rede, lógica de controle e funcionalidades necessárias com base em uma visão abstrata da rede. Essa programação ocorre no plano de aplicação, onde executam os softwares de gerenciamento [47]. Entre as aplicações comumente executadas nesse plano pode-se citar as de roteamento, monitoramento, detecção de intrusão, balanceamento de carga e *firewall* [48].

A arquitetura SDN facilita a implantação das políticas desejadas e torna a rede mais flexível a eventuais mudanças comportamentais, reduzindo os custos de gerenciamento [44]. Apesar dos benefícios, essa arquitetura também possui desvantagens. A centralização da inteligência da rede cria um ponto principal de falha. Deste modo, o controlador se torna vulnerável a, por exemplo, ataques de Negação de Serviço Distribuído [11], [45]. As redes SDN também são vulneráveis a outros tipos de ataques como: *scanning*, para obter informações sobre a rede alvo e possibilitar a execução de outros ataques; *spoofing*, onde os atacantes utilizam uma identidade falsa para obter privilégios ilegítimos na rede; *hijacking*, no qual se ganha total controle sobre o elemento de comunicação atacado; *tampering*, que se refere a um ataque onde informações da rede são alteradas; *man-in-the-middle*, no qual os dados do canal de comunicação entre o controlador e o plano de dados são interceptados e possivelmente alterados pelo agente malicioso [42], [43].

2.2 Anomalia de Rede

Na área de gerência de redes de computadores é comum a coleta de dados sobre o tráfego de rede a fim de efetuar análises de seu comportamento [49]. A maioria das observações coletadas tendem a seguir um certo padrão (*baseline*), calculado anteriormente com base na análise de dados históricos de tráfego [7], [14]. Observações que não seguem esse padrão podem ser chamadas de anomalias de rede [50], [51].

É ilustrado na Figura 3 um exemplo de anomalia em observações estatísticas genéricas. Há um conjunto de pontos (observações) dispostos em um plano cartesiano. Esses pontos são formados por coordenadas x e y (características). O modelo de comportamento normal de uma observação (*baseline*) é estar disposta próxima à reta traçada em azul. O ponto destacado em vermelho é uma anomalia, pois suas coordenadas desviam desse padrão e aumentam consideravelmente a sua distância da reta.

Fernandes *et al.* [15] categorizam as anomalias de rede conforme a sua natureza e causa. A primeira abordagem de categorização rotula uma anomalia como: pontual, quando um exemplo de dados isolado não segue o padrão de comportamento normal; coletiva, quando um conjunto de exemplos de dados têm um comportamento fora do padrão; contextual, quando um exemplo de dados se comporta de uma maneira específica em um contexto pré-determinado. A segunda forma de classificar anomalias considera quatro grupos distintos: operacional, que representa anomalias não maliciosas provenientes de

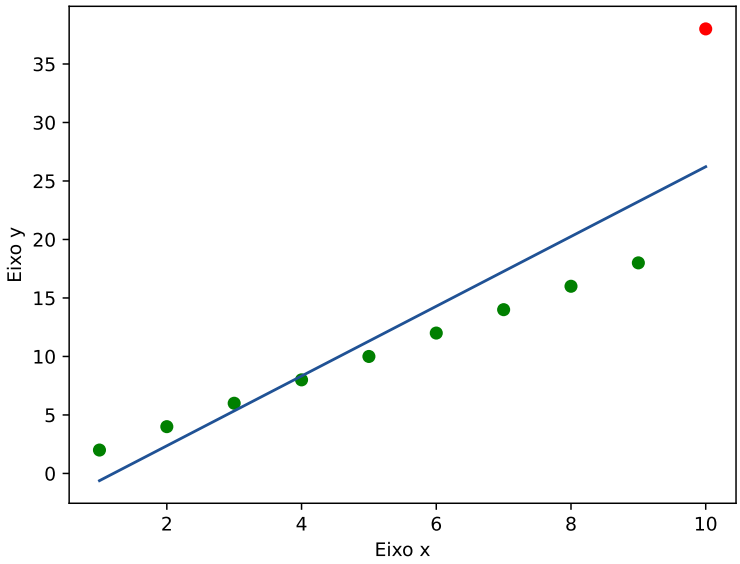

Figura 3 – Exemplo genérico de anomalia.

falha de hardware, software ou humana; *Flash Crowd*, que caracteriza um aumento súbito no volume de tráfego legítimo; medida, que configura falha na coleta dos dados de tráfego considerados na análise; e por fim, ataque de rede, que reflete ações de usuários maliciosos com o intuito de comprometer a confidencialidade, integridade ou disponibilidade de serviços de redes de computadores.

2.3 Sistema de Detecção de Intrusão de Redes

A ocorrência de anomalias no tráfego de uma rede de computadores pode comprometer a entrega dos seus serviços aos usuários, causando enormes prejuízos [12], [13]. As redes estão em constante evolução, assim como variações de ataques conhecidos e novos ataques são continuamente concebidos por agentes maliciosos [33]. Deste modo, a área de detecção de anomalias de redes tem sido estudada há muitos anos pela comunidade científica [34], [35], [36]. Pesquisas recentes destacam problemas nesse campo que ainda não possuem uma solução definitiva [37], [38], [15], [33], [25], [39], [40].

Um dos métodos mais aceitos para detectar anomalias de redes são os Sistemas de Detecção de Intrusão de Redes (NIDS, do inglês *Network Intrusion Detection System*). Esses sistemas são responsáveis por coletar e analisar o tráfego periodicamente. O NIDS alerta os gerentes de redes quando rastros anômalos forem identificados [30], como ilustrado na Figura 4. Ele age como uma segunda linha de defesa de redes contra ações maliciosas não capturadas pelos *firewalls*. Seu principal objetivo é preservar a confidencialidade, integridade e disponibilidade de redes de computadores [15].

É necessário identificar potenciais anomalias o mais rápido possível para que medidas de contenção possam ser tomadas antes do comprometimento da rede [52], [6]. O tamanho do intervalo de coleta e análise de tráfego é o parâmetro que dita o tempo de resposta de um NIDS a eventos anômalos. Pequenos períodos entre os momentos de análise possibilitam uma rápida identificação e mitigação de possíveis novos ataques lançados sobre a rede [29]. O tamanho desse intervalo reduziu-se ao longo do tempo devido à alta taxa de transmissão alcançada em redes de computadores atualmente [31], [11]. Por exemplo, uma rede que opera a uma velocidade de 100 Gbps pode ter 12,5 GBytes de dados comprometidos em apenas um segundo. Os sistemas propostos por [8] e [7] seguem esses princípios e efetuam a coleta e análise de observações de tráfego a cada segundo, rapidamente detectando e mitigando novos ataques à rede.

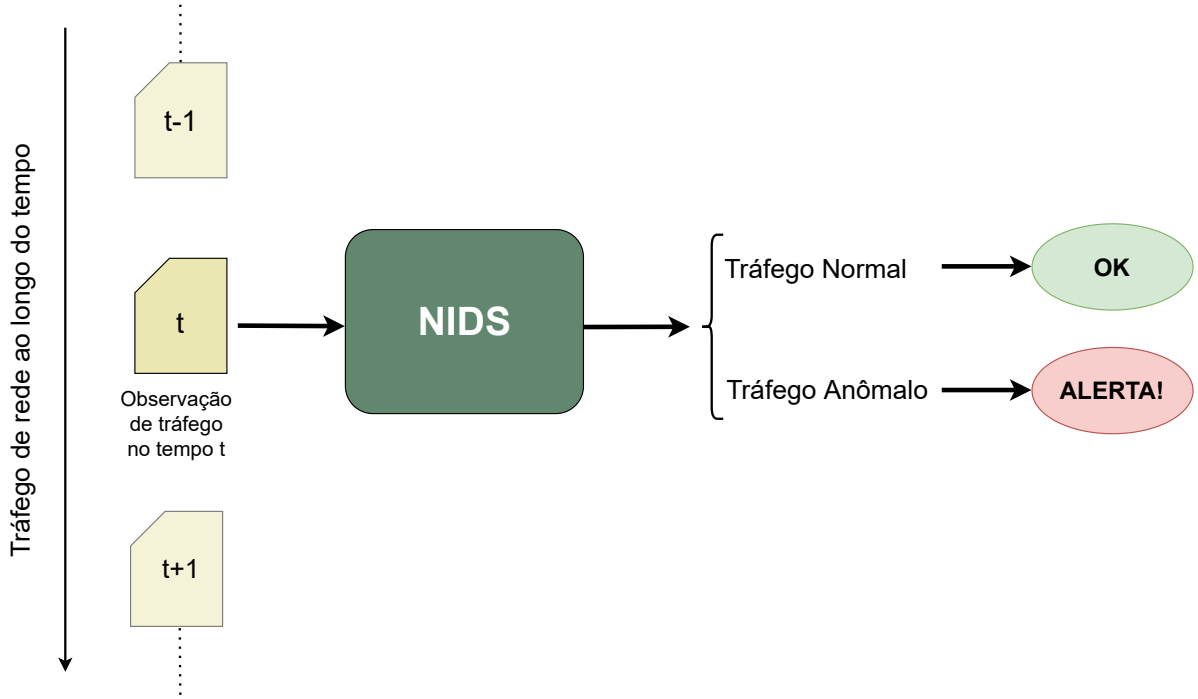

Figura 4 – Funcionamento de um NIDS.

Sistemas de Detecção de Intrusão de Redes podem ser classificados segundo a estratégia de detecção utilizada. As duas mais comuns são a baseada em assinatura e a baseada em anomalia [37]. Um NIDS implementado utilizando a primeira estratégia mantém um banco de dados contendo padrões de anomalias já conhecidos. O seu intuito é o de monitorar o tráfego de rede tentando identificar algum desses padrões anômalos nos dados [40]. Uma vantagem notável desse método é que a taxa de falsos positivos tende a ser baixa, já que o sistema só gera um alarme quando uma anomalia for realmente identificada [15]. A principal desvantagem dessa abordagem é que anomalias desconhecidas não são possíveis de serem identificadas, podendo causar uma alta taxa de falsos negativos [25]. Além disso, o banco de dados precisa ser constantemente atualizado. Pois, é comum o

surgimento de variações de anomalias conhecidas e de novas anomalias [30].

Já um NIDS baseado em anomalias constrói um modelo que representa o comportamento normal do tráfego utilizando dados históricos (*baseline*) [39]. O seu objetivo é monitorar o tráfego comparando o comportamento observado com o comportamento normal ou esperado. Uma anomalia é identificada quando a diferença entre o comportamento observado e o esperado ultrapassa um limiar pré-definido [25]. A vantagem desse método é que ele torna o sistema capaz de identificar tanto anomalias conhecidas quanto desconhecidas [53]. Por outro lado, a sua principal desvantagem é que esse tipo de abordagem tende a gerar uma alta taxa de falsos positivos [33]. A causa desse problema está no fato de que qualquer variação não maliciosa no comportamento do tráfego pode ser considerada uma anomalia. Essas variações são comuns, pois, o comportamento normal de uma rede pode mudar de tempos em tempos. Assim, é necessário que o modelo de comportamento normal seja atualizado constantemente e essa tarefa pode ter um custo elevado [15]. Esse tipo de NIDS é o mais utilizado e estudado pela comunidade científica e é a estratégia adotada no desenvolvimento do estudo de caso deste trabalho.

Existe um grande esforço em meio a comunidade científica na construção de Sistemas de Detecção de Intrusão de Redes que atendam a todas as necessidades de segurança. Deste modo, inúmeros trabalhos têm sido publicados ao longo dos anos. Um exemplo desse esforço pode ser observado no grupo de pesquisa ORION da Universidade Estadual de Londrina. Os pesquisadores que o compõem contribuem desde 2004 para o avanço da ciência nessa área do conhecimento [54] [55], [56], [57], [58], [14], [59], [11], [8]. Entre os seus diversos trabalhos publicados pode-se citar o produzido por Lent *et al.* [7], onde os autores descrevem um NIDS implementado com rede GRU e Lógica *Fuzzy*. Esse sistema foi desenvolvido para detectar e mitigar ataques a Redes Definidas por Software. Outro trabalho publicado pelo grupo foi proposto por Hamamoto *et al.* [59], que desenvolveram um NIDS baseado em anomalias autônomo e não-supervisionado. Esse sistema é implementado com o uso de Algoritmos Genéticos para a geração do modelo de comportamento normal. Além disso, foi utilizado Lógica Difusa no seu algoritmo de detecção de anomalias. Assis *et al.* [29] construíram um NIDS baseado em anomalias utilizando Rede Neural Convolucional para ambientes de Internet das Coisas. A comunidade científica usou diversos métodos para a implementação de um NIDS ao longo dos anos, como, por exemplo, Aprendizado de Máquina, Métodos Estatísticos, Teoria da Informação e Computação Evolucionária [60], [36], [15], [61], [62]. O método de Aprendizado de Máquina se destaca por obter, em geral, uma acurácia superior na resolução de problemas em comparação com modelos heurísticos e determinísticos [23], [24].

2.4 Aprendizado de Máquina

Existem tarefas para as quais é extremamente difícil especificar algoritmos que as resolvam, como, por exemplo, reconhecimento de fala [63]. Atualmente, existe uma enorme quantidade de dados que pode auxiliar na construção desses algoritmos. Um segundo exemplo de tarefa de difícil implementação é a de classificação de e-mails. Não há um algoritmo para determinar se um e-mail é spam ou não, mas existe uma abundância de exemplos de e-mails considerados spam ou legítimos. Uma solução para esses tipos de problemas é a aproximação automática de um algoritmo a partir dos dados disponíveis [64]. Esse tipo de solução é chamada de Aprendizado de Máquina (ML, do inglês *Machine Learning*), uma subárea de Inteligência Artificial (AI, do inglês *Artificial Intelligence*) [65], [25]. É ilustrada na Figura 5 a relação entre as subáreas de ML, onde se tem em destaque o subconjunto de Rede Neural, o qual contém um subconjunto especial chamado de Aprendizado Profundo.

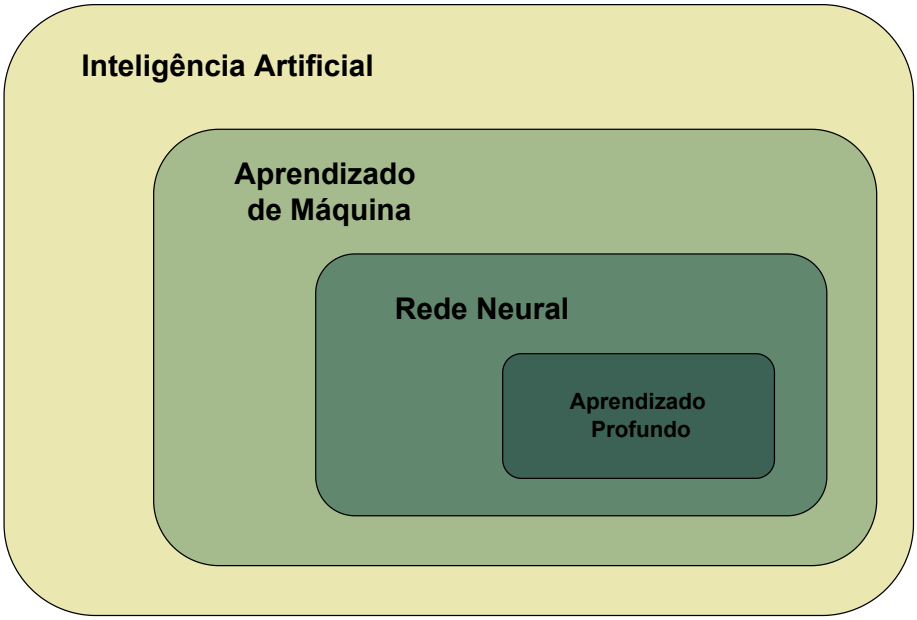

Figura 5 – Relação entre as áreas de aprendizado.

Aprendizado de Máquina é um método para o desenvolvimento de modelos computacionais que aprendem a performar uma tarefa sobre um conjunto de dados [66], [63]. Inicialmente, o sistema treina a realização da tarefa com o uso de um conjunto de dados específico. Durante o treinamento o sistema melhora automaticamente na realização da tarefa mediante experiência [67]. Após o treino, espera-se que o sistema possa executar a mesma tarefa para conjuntos de dados semelhantes que ele ainda não teve contato.

2.4.1 Tipos de Aprendizado

Os modelos de Aprendizado de Máquina podem ser classificados em três principais tipos: supervisionado, semi-supervisionado e não-supervisionado [33]. A principal diferença entre essas formas de aprendizado está presente nos dados utilizados no treinamento [68]. Modelos supervisionados utilizam em seu treinamento um conjunto de dados onde cada observação x possui um rótulo y que a descreve. O seu objetivo é aprender uma função f que mapeie um x qualquer para um y' que seja satisfatoriamente próximo do y real. Esse tipo de ML é utilizado para resolver problemas de classificação e de regressão. Entre os modelos de Aprendizado de Máquina utilizados pela comunidade científica pode-se citar Árvore de Decisão, Regressão Logística, Rede Neural e Máquina de Vetores de Suporte [67].

A tarefa de rotulação é extremamente custosa e com isso, geralmente, grande parte das observações dos conjuntos de dados disponíveis estão sem rótulo [69]. Modelos semi-supervisionados procuram utilizar observações com e sem rótulo no processo de treinamento. O seu objetivo é utilizar os dados não rotulados para construir um sistema supervisionado que seja melhor do que aquele construído utilizando somente os poucos dados rotulados disponíveis [63].

Modelos não supervisionados usam conjuntos de dados cujas observações não possuem rótulo. O objetivo básico desse tipo de modelo é encontrar padrões e estruturas que estão escondidas nos dados [53]. Eles são geralmente utilizados para resolver problemas de clusterização e redução de dimensionalidade. Exemplos desses modelos incluem Rede Neural, Agrupamento *k-means*, Análise de Componentes Principais [66].

2.4.2 Rede Neural

Rede Neural (NN, do inglês *Neural Network*) é um modelo de Aprendizado de Máquina que pode ser implementado para cada um dos tipos de aprendizado citados anteriormente [66]. Esse modelo é inspirado no cérebro humano e utiliza unidades chamadas de neurônios e suas interconexões para efetuar cálculos complexos [63]. Uma NN é formada por camadas sequenciais de neurônios que estabelecem conexões entre si, como ilustrado na Figura 6. A primeira camada recebe os dados de entrada. A última camada representa a saída y calculada para a entrada x alimentada na primeira camada. Entre as camadas de entrada e saída existem camadas ocultas. Deste modo, inúmeras variações de NN podem ser construídas ao alterar o número de neurônios em cada camada e o número de camadas ocultas [69].

2.4.3 Aprendizado Profundo

Aprendizado Profundo (DL, do inglês *Deep Learning*) representa um subconjunto de NN composto por Redes Neurais mais complexas. Essas redes possuem múltiplas ca-

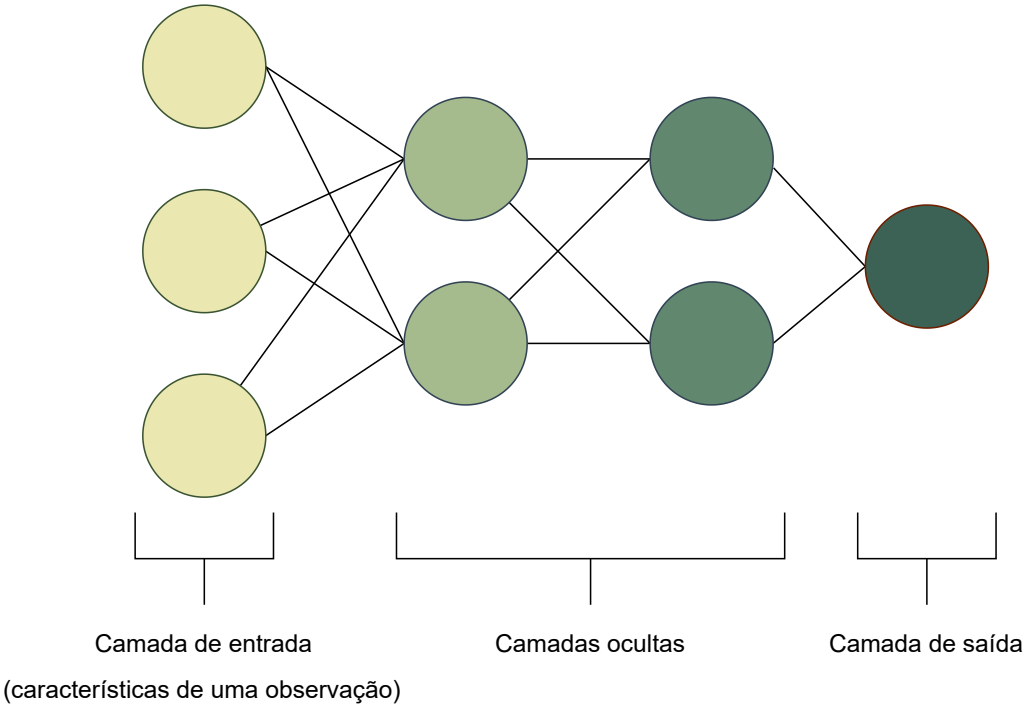

Figura 6 – Exemplo de Rede Neural.

madas ocultas, sendo chamadas de Redes Neurais Profundas (DNN, do inglês *Deep Neural Networks*) [68]. As DNNs têm ganhado popularidade nos últimos anos devido a sua aplicabilidade na solução de problemas em diversas áreas [70], [71]. Essa rede consegue modelar conceitos e funções bem mais complexas do que uma simples Rede Neural [65]. Como consequência, elas podem resolver problemas de complexidade superior.

Dentre as diferenças entre Aprendizado de Máquina e Aprendizado Profundo pode-se citar [33]: DL requer um grande volume de dados de treinamento, enquanto ML não; DL requer mais recursos computacionais do que ML; diferentemente de ML, os modelos de DL efetuam a engenharia de atributos (*feature engineering*) dos dados de maneira automática, reduzindo a intervenção humana. Essa é uma tarefa importante de manipulação do conjunto de dados original para um treinamento mais eficiente.

Os diferentes modelos de DNN podem ser classificados como discriminativos, generativos ou híbridos [30], [70]. Os primeiros seguem um treinamento supervisionado, como, por exemplo, a *Convolutional Neural Network* (CNN). Os generativos dizem respeito as redes cujo treinamento ocorre de maneira não supervisionada, enquanto os híbridos combinam os dois tipos de aprendizado. Exemplos de modelos generativos incluem *Autoencoder (AE)* e *Recurrent Neural Network (RNN)*. Um tipo de modelo híbrido é a *Generative Adversarial Network* (GAN), o qual é o foco de estudo deste trabalho.

2.4.3.1 *Long Short-term Memory*

A Rede Neural Recorrente (RNN, do inglês *Recurrent Neural Network*) é um modelo de Aprendizado Profundo. Ela é comumente utilizada em problemas cujos dados são sequencias e correlatos entre si, como, por exemplo, previsão de série temporal [7]. Diferentemente das redes neurais comuns, uma RNN possui neurônios especiais cuja saída é conectada na própria entrada. Isso implica que essa rede considera, para cada neurônio especial, as ativações calculadas previamente no cálculo das ativações atuais. Ou seja, os neurônios da rede RNN possuem memória sobre as instâncias de dados previamente processadas. Essa memória é considerada no processamento da instância de dados atual [5].

Uma RNN tradicional não consegue aprender dependências de longo prazo entre instâncias de dados, pois, a rede se lembra apenas de informações processadas em um passado próximo [72]. A rede falhará quando uma memória muito antiga for necessária para o processamento da instância atual. Esse ponto negativo existe devido ao problema da dissipação ou explosão de gradientes que ocorre durante o seu treinamento [73].

Uma variação da RNN tradicional foi proposta por Hochreiter *et al.* em 1997 [74] para solucionar esse problema, a *Long Short-Term Memory* (LSTM). Como ilustrado na Figura 7, uma LSTM possui neurônios especiais formados por mecanismos chamados de portões que regulam a memória da rede [28]. O portão f_t (equação 2.1) decide o quanto da memória de processamentos prévios C_{t-1} do neurônio será esquecida. O portão i_t (equação 2.2) decide o quanto da informação C'_t calculada atualmente (equação 2.3) será incorporada na memória. A atualização da memória é dada por C_t a partir da equação 2.4, que será utilizada no processamento da próxima instância de dados. Como descrito na equação 2.6, a ativação do neurônio h_t é calculada em função da nova memória C_t escalada com base no valor do portão o_t (equação 2.5). As funções σ e $tanh$ representam a *sigmoid* e a tangente hiperbólica, respectivamente.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{2.1}$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2.2}$$

$$C'_t = tanh(W_C[h_{t-1}, x_t] + b_C) \tag{2.3}$$

$$C_t = f_t * C_{t-1} + i_t * C'_t \tag{2.4}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.5}$$

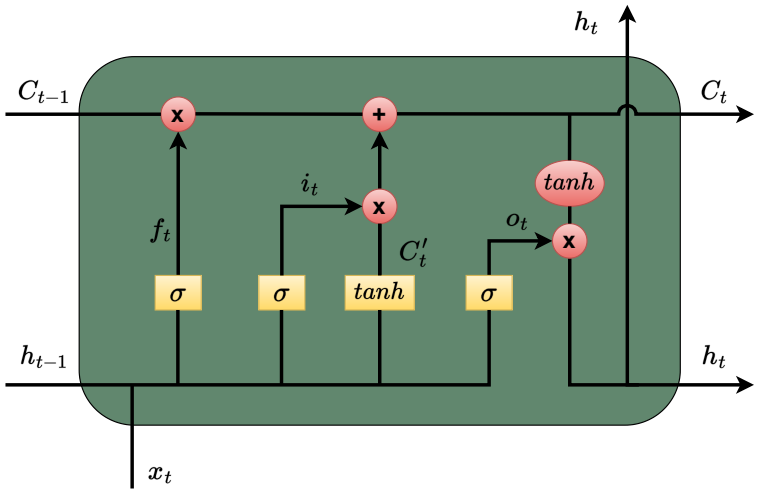

Figura 7 – Neurônio LSTM.

$$h_t = o_t * tanh(C_t) \tag{2.6}$$

2.4.3.2 *Gated Recurrent Unit*

Outra proposta de solução para o problema de aprendizado de dependências de longo prazo das RNNs foi introduzida por Cho *et al.* em 2014 [75], a *Gated Recurrent Unit* (GRU). Como ilustrado na Figura 8, o neurônio da rede GRU é semelhante ao da LSTM, porém utiliza menos portões. Essa característica reduz a quantidade de parâmetros da rede e com isso o custo computacional de treinamento [72].

Um neurônio GRU possui apenas dois portões [5]: r_t (equação 2.7) e u_t (equação 2.8). O primeiro é responsável por decidir o quanto da memória proveniente de computações anteriores h_{t-1} será considerada no cálculo da nova memória parcial h'_t do neurônio (equação 2.9). O segundo portão decide o quanto da memória passada h_{t-1} será esquecida. Além disso, ele também define o quanto do valor da nova memória parcial h'_t será somado a h_{t-1} de modo a atualizar a memória final h_t do neurônio (equação 2.10).

$$r_t = \sigma(W_r[h_{t-1}, x_t] + b_r) \tag{2.7}$$

$$u_t = \sigma(W_u[h_{t-1}, x_t] + b_u) \tag{2.8}$$

$$h'_t = tanh(W_h[r_t * h_{t-1}, x_t] + b_h) \tag{2.9}$$

$$h_t = (1 - u_t) * h_{t-1} + u_t * h'_t \tag{2.10}$$

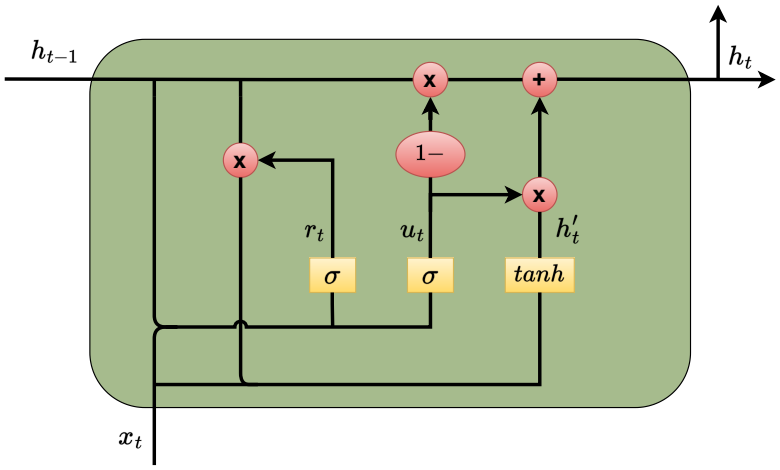

Figura 8 – Neurônio GRU.

2.4.3.3 *Convolutional Neural Network*

A Rede Neural Convolucional (CNN, do inglês *Convolutional Neural Network*) é um modelo de Aprendizado Profundo criado em 1989. Ele é aplicado principalmente no contexto de processamento de imagens [76]. Esse modelo também tem se mostrado eficiente para outros tipos de tarefa, como, por exemplo, previsão de série temporal e processamento de linguagem natural e fala. A sua arquitetura permite que ela identifique traços complexos nos dados de entrada utilizando menos parâmetros que uma Rede Neural Profunda convencional. Uma CNN é comumente formada por combinações de três tipos de camadas ocultas, como ilustrado na Figura 9: convolucional, *pooling*, e densa [30].

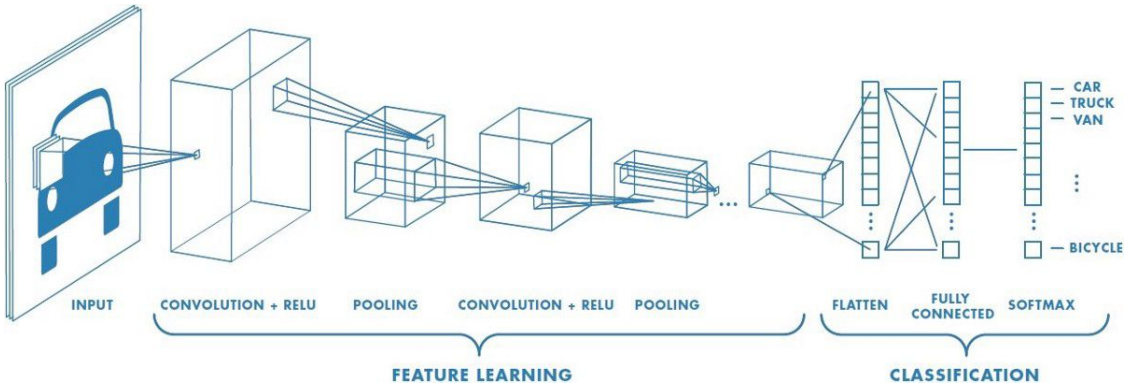

Figura 9 – Exemplo de Rede Neural Convolucional. (Fonte: 77)

A camada convolucional é formada por múltiplas matrizes de pesos chamadas de filtros [78]. Cada filtro é iterado pelo vetor multidimensional de dados de entrada produzindo um vetor de saída de dimensões reduzidas. Essas matrizes têm seus pesos ajustados automaticamente durante o treinamento da rede para identificar características especificas no vetor de entrada. No vetor de saída referente a um filtro específico tem-

se destacado os traços que essa matriz foi treinada para reconhecer. Por exemplo, no contexto de processamento de imagens, um filtro pode ser utilizado para detectar linhas horizontais na imagem de entrada. A imagem resultante da iteração do filtro na imagem original será composta majoritariamente por linhas horizontais. Essa camada geralmente utiliza a função de ativação ReLu (*Rectified linear unit*) para adicionar não-linearidade e controlar o intervalo de valores dos vetores de saída.

Uma camada de *pooling* é utilizada após uma camada convolucional, tendo como entrada os vetores resultantes da segunda. Ela visa reduzir ainda mais as dimensões desses vetores, de modo a diminuir os parâmetros de processamento [79]. Essa camada funciona iterando uma matriz por cada vetor. A cada iteração, é direcionando para outro vetor de saída correspondente o valor máximo entre os valores sobrepostos pela matriz.

Um modelo CNN geralmente possui múltiplas camadas convolucionais e de *pooling*. A cada camada convolucional, características mais e mais complexas do vetor de entrada original são identificadas pelos filtros [80]. Após o processamento de todas essas camadas, os vetores resultantes são agrupados em um único vetor unidimensional, que será a entrada de uma sequência de camadas densas.

As camadas densas (*fully-connected*) são semelhantes as das Redes Neurais comuns e terão o papel de produzir uma saída com base nas características espaciais dos dados de entrada identificadas pelas camadas anteriores [70]. Para um problema de classificação de imagens, por exemplo, pode-se ter uma camada densa de saída onde cada neurônio representa uma das possíveis classes que a imagem pode se enquadrar.

2.4.3.4 *Temporal Convolutional Network*

A Rede Convolucional Temporal (TCN, do inglês *Temporal Convolutional Network*) é uma variação da CNN formalizada em 2018 por Bai *et al.* [81]. Esse tipo de rede neural é utilizada para problemas de modelagem de sequência. A rede TCN tem se mostrado superior às suas concorrentes LSTM e GRU em alguns tipos de tarefas, como, por exemplo, predição de série temporal [82], [83], [84].

O principal componente de uma Rede Convolucional Temporal é a convolução causal dilatada. Ela é utilizada para extrair padrões específicos nos dados conforme os filtros aprendidos [85]. Como ilustrado na Figura 10, essa convolução é implementada utilizando n camadas de mesmo tamanho da entrada formadas por filtros de tamanho k. Um *padding* de tamanho $k-1$ é adicionado à entrada da convolução para manter o tamanho da entrada e saída iguais e obter a propriedade casual. Isto é, uma saída produzida no tempo t é fruto da convolução de entradas observadas no tempo t e em tempos anteriores. Cada camada deste tipo de convolução possui um parâmetro associado a ela chamado de fator de dilatação (d). Ele indica a distância entre os componentes dos filtros da camada. Deste modo, o campo receptivo da rede pode ser expandido ao aumentar

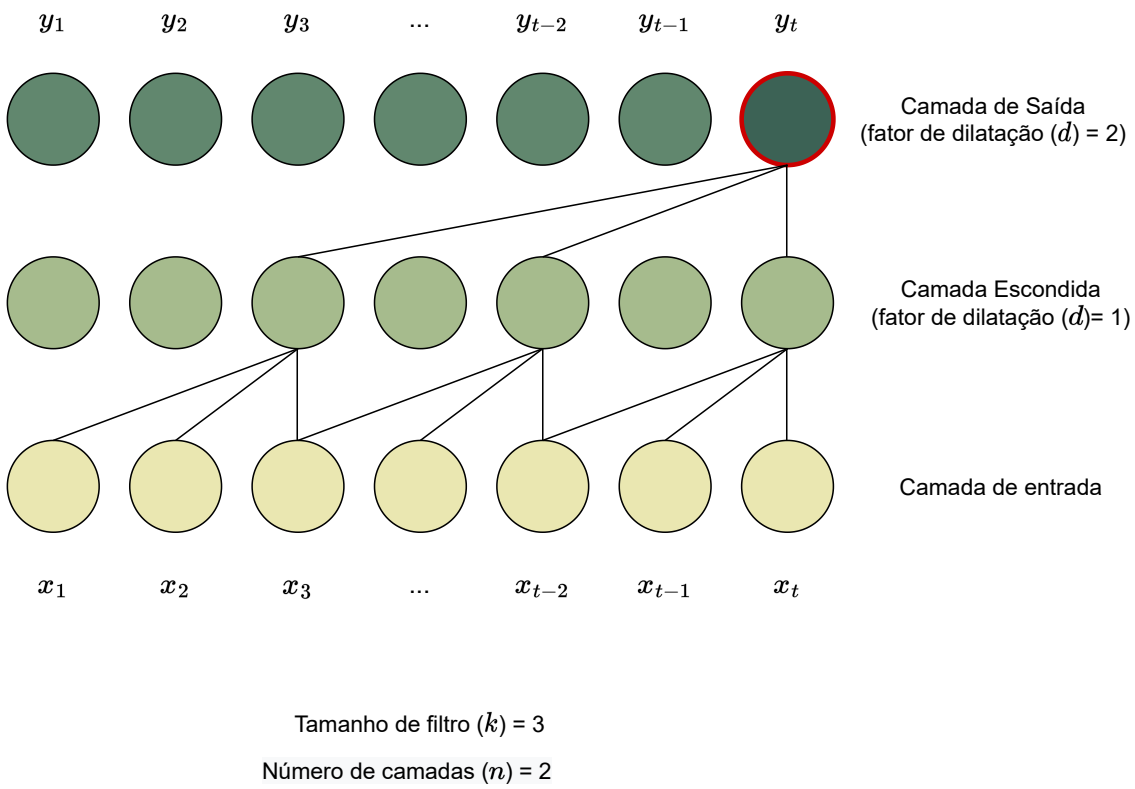

Figura 10 – Exemplo de convolução causal dilatada.

o número n de camadas da convolução, o tamanho de filtro k e o fator de dilatação d [84].

Uma TCN formada por convoluções causais dilatadas que possuem muitas camadas internas pode sofrer do problema de dissipação ou explosão de gradientes durante o treinamento [82]. Para resolver esse problema o componente básico de uma arquitetura TCN passa a ser o bloco residual, como apresentado na Figura 11. Esse bloco é formado por dois ramos que recebem a entrada de dados: um composto por transformações e um para performar uma convolução unidimensional [81]. O primeiro é formado por duas partes sequenciais que possuem: uma camada causal dilatada para realizar a extração de características dos dados, uma camada de normalização, uma função de ativação para adicionar não linearidade ao processo e por fim uma camada de regularização. O resultado do segundo ramo é somado ao do primeiro para garantir que a entrada e a saída do bloco possuam o mesmo tamanho.

Observou-se um padrão para a construção de arquiteturas TCN com base em blocos residuais [86], [82], [83]: N blocos residuais para realizar a extração de padrões dos dados seguidas por camadas densas que os mapeiam para uma saída adequada ao problema sendo resolvido.

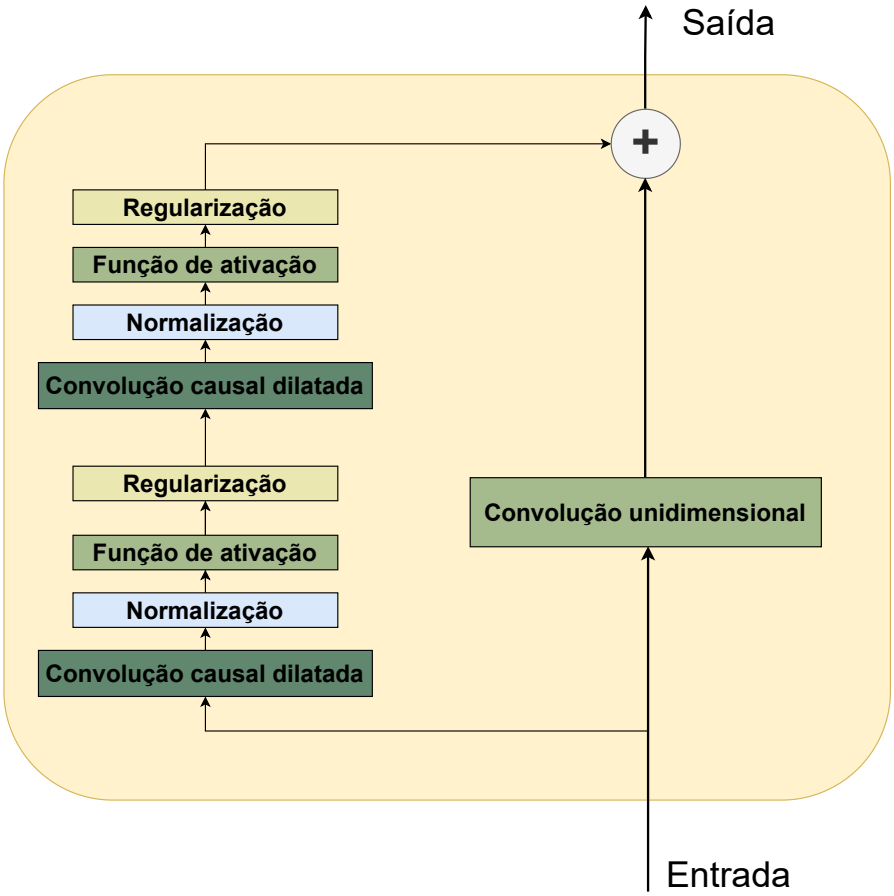

Figura 11 – Arquitetura de um bloco residual.

2.4.3.5 Comparação dos Modelos

A tabela 1 descreve as principais características de cada um dos quatro modelos apresentados anteriormente. A princípio, pode-se notar que todos os modelos são aplicáveis no contexto do problema de modelagem de sequência. Apesar das redes LSTM e CNN terem sido concebidas no século passado, elas ainda possuem um grande valor atualmente. As redes GRU e TCN foram formalizadas mais recentemente e têm sido amplamente exploradas em tarefas de modelagem de sequência. Ainda não existe um consenso na comunidade científica sobre qual modelo é o melhor. Diversos trabalhos estudam as vantagens e desvantagens de cada um deles em diferentes cenários de testes, como, por exemplo, [82], [83], [84], [76].

Tabela 1 – Modelos de Aprendizado Profundo.

Modelo	**Ano**	**Tipo de aprendizado**	**Descrição**	**Aplicação comum**
LSTM	1997	Supervisionado	É uma Rede Neural Recorrente com neurônios especiais que armazenam memória sobre instâncias de dados processadas previamente (tempo $t-1$ ou antes). Essa memória influencia no processamento do exemplo de dados atual (tempo t). A LSTM é capaz aprender dependências de longo prazo entre diferentes observações de dados.	Modelagem de sequência (por exemplo, previsão de série temporal)
GRU	2014	Supervisionado	É uma Rede Neural Recorrente assim como a LSTM e possui as mesmas características básicas. Sua diferença se encontra na arquitetura de seu neurônio especial, que utiliza menos parâmetros.	Modelagem de sequência
CNN	1989	Supervisionado	É tipo de Rede Neural que utiliza menos parâmetros que DNNs convencionais. A CNN extrai características espaciais complexas de exemplos de dados e as utiliza para classificá-los.	Processamento de imagem e previsão de série temporal.
TCN	2018	Supervisionado	Assim como a rede CNN, essa rede é baseada na ideia de camadas convolucionais. Essas camadas extraem características complexas dos dados. Posteriormente, elas são mapeadas para a saída adequada ao problema. A diferença é que a rede TCN é desenvolvida para trabalhar especificamente com dados sequências. Além disso, o processamento realizado no tempo t só pode ser influenciado por informações observadas até t. Ademais, diferentemente da CNN, as suas saídas são sempre iguais às entradas. Camadas de *pooling* não são utilizadas nesse modelo.	Modelagem de sequência

2.4.4 Treinamento e Hiperparâmetros

Redes neurais são modelos computacionais desenvolvidos para aprenderem a realizar uma certa tarefa sobre um conjunto de dados. Esse desenvolvimento é composto por duas fases, como apresentado na Figura 12: treinamento e teste. O conjunto de dados considerado é dividido em três partes: treino, validação e teste. Os dois primeiros são utilizados na fase de treinamento e o terceiro na fase de teste [64]. Na primeira fase, o modelo aprende a performar a tarefa com base nas duas partições de dados alocadas. A eficiência do modelo na realização da tarefa é avaliada na segunda fase considerando uma parcela de dados que não foi utilizada em seu treinamento [68]. Essa avaliação é baseada em métricas estatísticas de desempenho, como, por exemplo, *F1-score* e *Matthews Correlation Coefficient* (MCC) [87], [88].

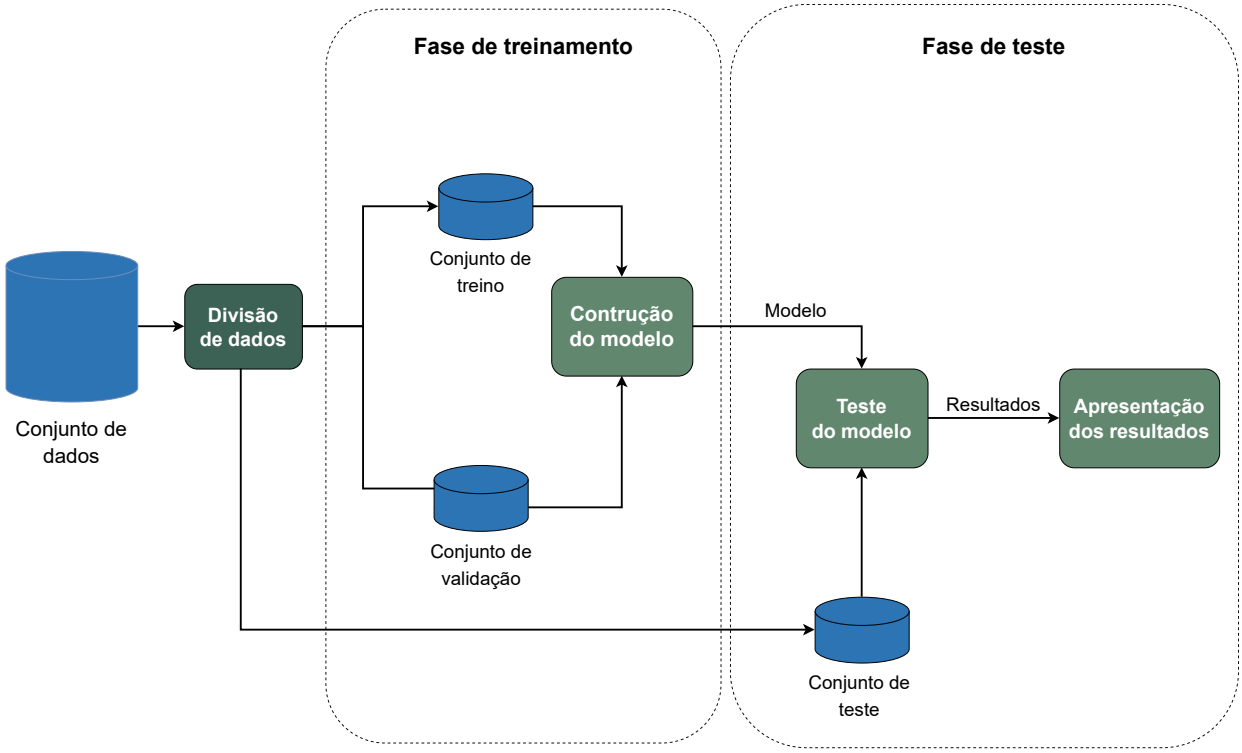

Figura 12 – Processo de desenvolvimento de uma rede neural.

O conhecimento de uma rede neural é representado por variáveis internas ao modelo, chamadas de parâmetros (pesos e vieses) [63]. Na fase de treinamento, o conjunto de treino é utilizado para ajustar automaticamente esses parâmetros. Os ajustes são feitos com base na função de perda. Essa função recebe como entrada os parâmetros que compõem o modelo atualmente e calcula o seu erro na execução da tarefa. O treinamento da rede resume-se a derivar iterativamente a entrada (parâmetros) que reduz o valor da função. Ao final desse processo obtém-se a entrada que está associada ao mínimo global da função de perda [65]. Assim, os parâmetros internos que minimizam o erro do modelo são encontrados e a tarefa é performada com uma boa acurácia. Esse procedimento é

ilustrado de maneira simplificada na Figura 13. O ponto preto indica o erro associado aos parâmetros iniciais do modelo. A linha representa os erros dos diferentes conjuntos de parâmetros encontrados ao longo do treinamento. A seta aponta a direção dos parâmetros ótimos derivados ao final do treinamento.

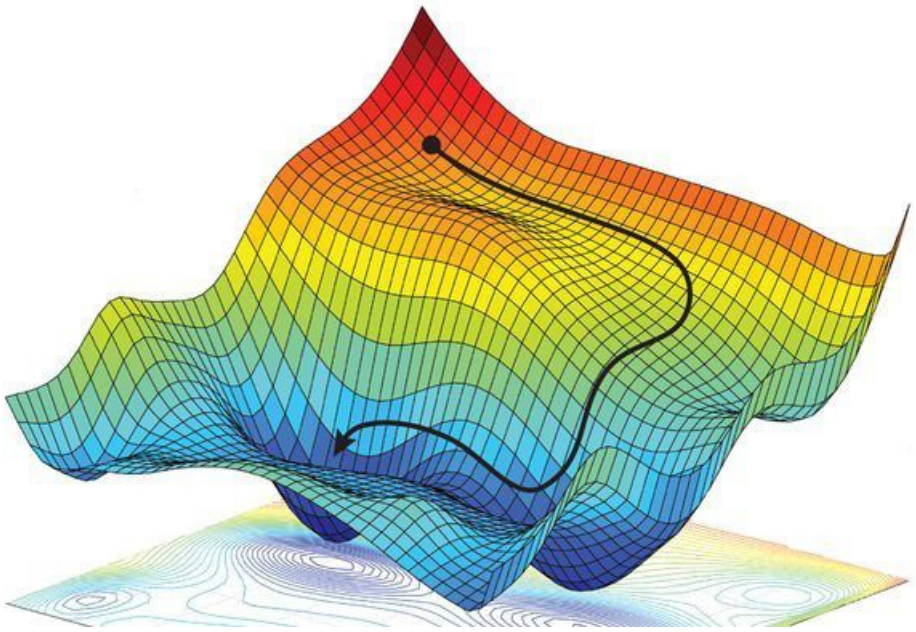

Figura 13 – Processo de cálculo do mínimo global de uma função. (Fonte: 89)

Existem outros parâmetros cujos valores, diferentemente dos anteriormente citados, não podem ser encontrados a partir de treinamento. Essas variáveis são chamadas de hiperparâmetros e têm influência direta na qualidade de treinamento da rede [90], [91]. Os hiperparâmetros configuram a arquitetura da rede neural e características de seu treinamento. A escolha adequada dessas variáveis aumenta a capacidade de aprendizado do modelo e a seu desempenho na resolução do problema que está sendo considerado, como ilustrado nos experimentos propostos em [7].

Toda rede neural possui hiperparâmetros básicos como [92], [93]:

- **Número de camadas ocultas e de neurônios em cada camada**: definem a arquitetura da rede neural. Um valor inadequado para esses parâmetros pode aumentar desnecessariamente a complexidade do modelo e causar o problema de *overfitting*. Esse problema refere-se à situação onde o modelo tem um bom desempenho para os dados de treinamento. Porém, não consegue generalizar os seus padrões, obtendo uma baixa acurácia na análise dos dados de teste [70].

- **Quantidade de épocas**: refere-se ao total de iterações sobre todo o conjunto de treino realizadas durante a fase de treinamento. Essa variável também pode ocasi-

onar *overfitting* caso assuma um valor muito elevado.

- **Tamanho de *mini-batch***: para cada iteração de treinamento (época), o conjunto de treino é embaralhado e dividido em subconjuntos *mini-batches* de um tamanho específico. Considerando cada *mini-batch* individualmente, atualizam-se levemente os parâmetros internos da rede visando alcançar gradualmente o mínimo da função de perda.

- **Taxa de aprendizagem**: as atualizações realizadas para cada *mini-batch* podem receber um peso que reduz ou mantém o seu valor, chamado de taxa de aprendizagem.

- **Taxa de *dropout***: dita a proporção de neurônios aleatórios desativados durante o treinamento. Essa é uma técnica de regularização implementada para dificultar o aprendizado do modelo e reduzir as chances de *overfitting*.

- **Função de ativação**: é uma função não linear incluída em cada neurônio da rede para permitir que relações complexas possam ser modeladas pela rede neural. A saída que seria produzida pelo processamento de um neurônio é tomada como entrada da função de ativação, produzindo uma nova saída não linear.

- **Função de custo**: recebe como entrada os parâmetros que compõem o modelo atualmente e calcula o seu erro na execução da tarefa em questão.

- **Otimizador**: existem diferentes implementações do algoritmo de treinamento de redes neurais, chamados de otimizadores. Como já mencionado, esses algoritmos buscam encontrar o mínimo global da função de custo associada ao modelo. Exemplos de otimizadores incluem *Stochastic Gradient Descent* (SGD), *Adaptive moment estimation* (Adam) e *Root Mean Square propagation*.

Algumas Redes Neurais possuem hiperparâmetros exclusivos devido às suas características específicas. Por exemplo, a CNN possui variáveis como tamanho de filtro, de *stride* e de *padding* [76], [78]. A rede TCN tem o tamanho de filtro da convolução e fator de dilatação como hiperparâmetros utilizados para variar o tamanho do seu campo receptivo [84].

O conjunto de validação é utilizado pelo desenvolvedor na fase de treinamento para auxiliá-lo a encontrar os valores ideais para os hiperparâmetros da rede [94]. Os valores ótimos dessas variáveis podem ser encontrados manualmente por meio de experimentos.

Essa estratégia pode requerer um grande tempo de execução. Por outro lado, o desenvolvedor também pode optar por automatizar esse processo utilizando métodos como *Grid Search* e *Random Search* [95]. A sua desvantagem é que eles tendem a exigir um alto custo computacional [96].

Por fim, a fase de teste visa simular a aplicação do modelo desenvolvido em um ambiente real onde os dados são desconhecidos. Para isso, a rede neural treinada e com os hiperparâmetros ajustados é utilizada no processamento do conjunto de teste. Esse conjunto possui dados nunca analisados pelo modelo. Espera-se que a rede aprenda a generalizar os padrões presentes nos dados de treinamento [97]. Assim, ela terá um bom desempenho no conjunto de teste na realização da tarefa em questão com base em métricas estatísticas.

3 REDE ADVERSÁRIA GENERATIVA

A Rede Adversária Generativa (GAN, do inglês *Generative Adversarial Network*) representa um modelo de Aprendizado Profundo concebido no ano de 2014 por *Goodfellow et al.* [98]. Esse tipo de DL é baseado na Teoria dos Jogos e é composto por duas Redes Neurais internas que competem entre si, o gerador e o discriminador, como ilustrado na Figura 14 [32]. O gerador aprende a modelar a distribuição dos dados de treinamento e gerar exemplos de dados sintéticos que se encaixam nessa distribuição. Para realizar a geração, a rede geradora recebe como entrada um vetor de ruído aleatório (z). Esse vetor é extraído do chamado espaço latente, composto por dados que seguem uma distribuição bem definida, como, por exemplo, a uniforme ou a Gaussiana. Já o discriminador representa um classificador binário que produz probabilidades como saída. Ele é treinado para discernir exemplos falsos ($G(z)$), criados pelo gerador, dos exemplos reais de dados (x) [99].

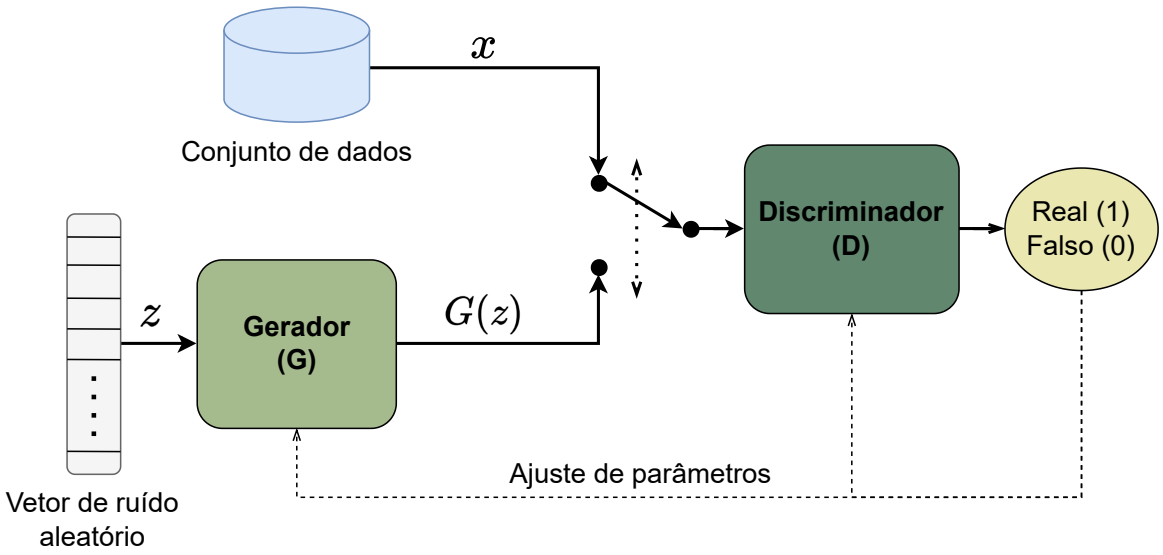

Figura 14 – Arquitetura da Rede Adversária Generativa.

O processo de treinamento torna cada uma das duas redes progressivamente melhores na execução de suas respectivas tarefas. O propósito final da rede GAN é atingir o Equilíbrio de Nash. Esse equilíbrio é representado pelo conjunto de parâmetros ótimos para as redes. Esses parâmetros fornecem o valor mínimo possível para as funções de custo do gerador e do discriminador [100]. Quando esse objetivo é atingido obtém-se um gerador que aprendeu a distribuição dos dados e pode gerar exemplos muito parecidos com os reais. A rede geradora consegue enganar totalmente a discriminadora. Ou seja, a segunda se torna incapaz de indicar se exemplos gerados são sintéticos ou reais. Deste

modo, a rede produz probabilidades próximas de 0,5, indicando a sua incerteza quanto a classificação das entradas [101], [32].

3.1 Processo de Treinamento

Uma rede GAN é treinada de maneira semelhante às redes tradicionais, utilizando como base o algoritmo de gradiente descendente [102]. Durante n iterações (épocas) o conjunto de dados é embaralhado e dividido em subconjuntos de tamanho m. Para cada subconjunto (*mini-batch*), com base nos exemplos de dados que os compõem, os parâmetros do discriminador são ajustados k vezes e os do gerador uma única vez. Esses ajustes acontecem separadamente para cada rede interna. Assim, enquanto os parâmetros do discriminador são modificados, os do gerador são mantidos fixos, e vice-versa [98].

Essa atualização de parâmetros ocorre com base na função de custo expressa por 3.1 [103]. O gerador visa minimizar a função. Isto é, espera-se que os valores de $D(x)$ sejam próximos de 0 enquanto os de $D(G(z))$ sejam próximos de 1. Assim, o discriminador estaria sendo enganado a classificar um exemplo de dados real como sendo falso e um exemplo de dados sintético como sendo real. Por outro lado, o propósito do discriminador é maximizar essa função. Ou seja, deseja-se aproximar os valores de $D(x)$ de 1 e os de $D(G(z))$ de 0. Dessa forma, o discriminador estaria classificando as suas entradas corretamente [104].

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{3.1}$$

Os parâmetros do discriminador são modificados utilizando o algoritmo de gradiente ascendente já que se objetiva a maximização da função 3.1. Por outro lado, considerando que o intuito do gerador é a minimização dessa mesma função, os seus parâmetros são alterados com base no algoritmo de gradiente descendente [32].

O processo de treinamento de uma rede GAN é uma instância do problema do jogo de soma zero. Uma das redes internas melhora o seu desempenho reduzindo o da adversária na mesma proporção [52]. Para todos os jogos desse tipo existe um certo momento onde nenhum dos dois jogadores consegue melhorar a sua situação, o chamado Equilíbrio de Nash [105]. É nesse momento em que o treinamento da rede GAN se encerra e que o gerador consegue aprender a distribuição dos dados originais. Neste caso, o discriminador se torna incapaz de distinguir os exemplos de dados.

3.2 Variações

A proposta original de Rede Adversária Generativa possui problemas, como por exemplo *mode collapse*, *mode drop*, instabilidade no processo de treinamento e dissipação

de gradientes [32]. O primeiro problema diz respeito à falta de diversidade nos exemplos gerados pela rede geradora [104]. O segundo ponto de falha acontece quando o gerador ignora um certo subconjunto de exemplos e não aprende a gerá-los por ter dificuldades de realizar a sua modelagem [106]. Já o terceiro refere-se à dificuldade de atingir o Equilíbrio de Nash, pois, na prática, o treinamento da rede GAN é instável [101]. O quarto problema ocorre quando o aprendizado do discriminador evolui mais rapidamente do que o do gerador. A rede discriminadora tende a rotular os exemplos de dados corretamente, minimizando os gradientes do gerador e impedindo-o de evoluir [103]. Assim, surgiram diversas variações de GAN que objetivam corrigir as dificuldades do modelo inicial.

A variação *Deep Convolutional* GAN foi proposta com o intuito de aplicar eficientemente a rede GAN no processamento de imagens [52]. Tanto o discriminador quanto o gerador são construídos com base em redes CNN [107]. Essas redes são formadas apenas por camadas convolucionais que contêm normalização de lote. Essa forma de implementar a rede GAN garante mais estabilidade durante o seu treinamento, evitando o problema do *mode collapse*.

A *Conditional* GAN é uma extensão da GAN original que adiciona uma nova variável de entrada (c) ao gerador e ao discriminador [101]. Essa entrada controla qual tipo de dados será gerado pelo gerador e analisado pelo discriminador. Diferentes valores de c fazem com que diferentes tipos de dados sejam gerados. Essa alteração na arquitetura da rede evita o *mode collapse*.

Para resolver o problema da dissipação de gradientes e de falha de convergência foi proposto o modelo *Wasserstein* GAN [52]. Essa variação não altera a arquitetura da rede, como as duas anteriores, mas sim a função de custo utilizada no seu treinamento. Ela utiliza a distância *Earth-Mover* na sua implementação. Existe uma extensão da própria *Wasserstein* GAN, a chamada *Wasserstein GAN with Gradient Penalty*, que resolve problemas encontrados posteriormente nessa arquitetura [108].

3.3 Métricas de Avaliação

Uma forma básica de mensurar o desempenho de modelos generativos como a GAN é medindo a dissimilaridade da distribuição aprendida pelo gerador e a distribuição real dos dados [32]. Boas métricas de desempenho devem ser sensíveis a *overfitting, mode collapse* e *mode drop* e penalizar modelos que apresentem esses problemas. Ao mesmo tempo, elas devem favorecer modelos que gerem exemplos muito próximos aos reais e que possuam diversidade entre si [106]. Além disso, essas métricas devem requerer baixo custo computacional de cálculo de modo que possam ser utilizadas de maneira prática.

Ainda não existe uma métrica definitiva aceita pela comunidade científica para avaliar as redes GAN. Esse é um ponto em aberto estudado nos últimos anos [101]. A

métrica mais simples definida é a de inspeção visual dos resultados realizada por um agente humano. Porém, como já apontado por Li *et al.* [99] e Xu *et al.* [109], essa estratégia é enviesada, não confiável e de alto custo. Assim, o foco de estudo é direcionado a métricas quantitativas, como, por exemplo: *Mode Score*, que avalia a qualidade e diversidade das imagens produzidas pelo gerador; *Fréchet Inception Distance*, que fornece a distância entre a distribuição de dados original e a gerada; *Kernel Maximum Mean Discrepancy*, que calcula a dissimilaridade entre a distribuição dos dados gerados (P_g) e a distribuição dos dados reais (R_r); e *1-Nearest neighbour classifier*, que quantifica o grau de similaridade entre as duas [103].

Xu *et al.* [109] realizaram um estudo empírico para avaliar adequabilidade de diferentes métricas na avaliação de modelos generativos. Os autores concluíram que as métricas *Kernel Maximum Mean Discrepancy* e *1-Nearest neighbour classifier* satisfazem a maioria das propriedades necessárias citadas anteriormente e que, portanto, são boas escolhas de medida de desempenho. Segundo Borji *et al.* [106], não há uma única métrica que avalie todos os aspectos do processo generativo. Deste modo, uma forma mais adequada de mensurar o desempenho de GANs seria utilizar mais de uma métrica para cobrir os diversos pontos a serem avaliados.

3.4 Aplicações

3.4.1 Geração de Dados

Como GAN é um tipo de modelo generativo, uma de suas principais aplicações é a de geração de dados [101]. O trabalho publicado por Waheed *et al.* [110] recorre à rede CNN para detectar a infecção por COVID-19 em pacientes com base em imagens raio-x da região peitoral. Os autores destacam que a falta de imagens para efetuar o treinamento da rede reduz o seu desempenho de detecção. Eles utilizaram uma rede GAN para gerar imagens sintéticas e aumentar o conjunto de dados de treinamento da rede. Huang *et al.* [111] e Ding *et al.* [112] utilizaram GAN para gerar exemplos de dados sintéticos e resolver o problema de desbalanceamento de classes. Esse problema prejudica o desempenho dos algoritmos de Aprendizado de Máquina e precisa ser tratado para garantir um treinamento de qualidade [113].

3.4.2 Processamento de Imagens

Outra área de aplicação na qual as redes GAN têm tido bons resultados é a de processamento de imagens [101]. Din *et al.* [114] utilizaram essas redes na construção de um sistema que tem como entrada uma imagem de uma pessoa utilizando máscara. Após o processamento, é produzido como saída uma imagem próxima da realidade dessa mesma pessoa sem a máscara no rosto. Zhang *et al.* [115] usaram uma *Stacked Generative*

Adversarial Network para construir um sistema que gera imagens fotorrealistas. Slam *et al.* [116] aplicaram *Conditional* GAN para melhorar a qualidade de imagens subaquáticas capturadas por robôs de exploração marítima em tempo real. Li *et al.* [99] destacam a aplicabilidade das redes GAN na tarefa de super-resolução de imagens. Esse tipo de solução visa transformar uma imagem de baixa resolução em uma imagem de alta resolução. Kumar *et al.* [103] apresentaram uma revisão da literatura a respeito das aplicações e desafios da rede GAN. Entre as aplicações destacadas no domínio de processamento de imagens estão: diagnosticação de imagens médicas; segmentação de imagens; aprimoramento de imagens; sintetização de faces humanas.

3.4.3 Áudio e Vídeo

Jabbar*et al.* [102] realizaram revisão da literatura a respeito das redes GAN. Duas das aplicações destacadas pelos autores são a de síntese de áudio e vídeo. Entre os tipos de áudio sintetizados tem-se música e poesia. No domínio de vídeo, os pesquisadores apresentaram aplicações para geração de vídeos de alta qualidade. Além disso, menciona-se a utilidade da GAN na tarefa de prever os movimentos de objetos.

3.4.4 Processamento de Linguagem Natural

Wang *et al.* [104] e Cheng *et al.* [108] apresentaram uma revisão dos trabalhos que aplicam a Rede Adversária Generativa. Ambas as pesquisas avaliaram a aplicação de GAN no contexto de Processamento de Linguagem Natural. Recentemente a rede GAN obteve bons resultados nesse domínio de problemas. Os autores citaram vários trabalhos desenvolvidos para esse tipo de processamento. Dentre as soluções introduzidas por esses trabalhos inclui-se: geração de texto a partir de diálogos; sintetização de sentenças textuais realistas; aprimoramento e geração de fala; produção de imagens a partir de texto.

3.4.5 Detecção de Anomalias

A rede GAN é um tópico de pesquisa relevante que demanda a atenção da comunidade científica nos últimos anos. Recentemente, essa rede é aplicada na área de detecção de anomalias, principalmente para aprender o perfil de comportamento normal dos dados [52]. Lee *et al.* [117] estudam e comparam diferentes sistemas de detecção de anomalias baseados em Redes Adversárias Generativas. Schlegl *et al.* [118] propõem um sistema não-supervisionado para a detecção de anomalias em imagens de retina. Para a sua implementação foi utilizado uma *Wasserstein* GAN juntamente com um *Encoder*. Como será apresentado na próxima seção, a rede GAN também têm se mostrado aplicável no desenvolvimento de Sistemas de Detecção de Intrusão de Redes.

4 TRABALHOS RELACIONADOS

A área de detecção de anomalias e intrusões de redes de computadores tem sido amplamente estudada pelos cientistas nos últimos anos [34], [35], [36]. Entre os modelos para a implementação desses sistemas, os que tem se destacado devido aos seus resultados promissores foram os de Aprendizado Profundo [26], [27], como, por exemplo, DNN, CNN, TCN, AE, RNN e GAN.

Vinayakumar *et al.* [19] realizaram um estudo sobre o desempenho de diferentes modelos de Aprendizado de Máquina na implementação de Sistemas de Detecção de Intrusão. Após múltiplos experimentos, os autores concluem que, de fato, um modelo de Aprendizado Profundo como a *Deep Neural Network* (DNN) apresenta resultados superiores nas métricas consideradas. Deste modo, utilizando como base uma DNN, os autores propõem um Sistema de Detecção de Intrusão híbrido capaz de identificar ataques ao nível de rede e ao nível de *host*. O sistema é implementado de maneira distribuída e escalável. Isso possibilita que ele consiga analisar em tempo real o grande volume de dados gerado pelas redes. Os autores destacam a importância do sistema ser adaptável à dinamicidade e complexidade observada atualmente nas redes de computadores. Entre os conjuntos de dados selecionados para a realização dos experimentos estão KDDCup99, NSL-KDD, UNSW-NB15, Kyoto, WSN-DS e CICIDS2017. Ademais, para avaliar o desempenho de detecção de anomalias do sistema foram utilizadas métricas como *Accuracy, Precision, F1-score, True Positive Rate, False Positive Rate* e *ROC curve*.

Khan *et al.* [21] desenvolveram um NIDS baseado em *Deep Stacked Autoencoder* aplicado a ambientes de redes de computadores tradicionais. O sistema é composto de duas etapas que executam de maneira serial. Ambas são implementadas utilizando um *Deep Stacked Autoencoder* para realizar a tarefa de *feature extraction* e um classificador *softmax* para performar a tarefa de rotulação. Na primeira etapa realiza-se a extração de características (*features*) e a classificação binária da instância de dados de entrada, cujas possíveis classes são normal e anômalo. Em seguida, na etapa dois, o resultado da classificação é concatenado com a entrada de dados original, processado por um segundo *Deep Stacked Autoencoder* e rotulado por um novo classificador *softmax*. Nesta etapa executa-se a tarefa de classificação multi-classe, assim, o exemplo de dados recebe um entre muitos possíveis rótulos. O usuário tem a opção de programar o sistema para executar apenas a etapa um ou executar ambas sequencialmente. Os autores destacam a importância da redução de características dos dados originais para obter um sistema que performa mais eficientemente em ambientes reais. O NIDS apresentou uma Acurácia de 99,996% e 89,134% nas bases de dados públicas KDDCup99 e UNSW-NB15, respectivamente. Em comparação com outros modelos do estado da arte, como *Random Forest,*

Logistic Regression, e *Decision Tree*, o dos autores obtém resultados superiores e utiliza menos atributos de dados.

ElSayed *et al.* [27] propuseram um estudo sobre o desenvolvimento de NIDS baseado em uma CNN combinada com diferentes classificadores e técnicas de regularização para proteger ambientes SDN. A rede CNN tem o papel de realizar a extração de características (*features*) de alto nível dos dados. Na sequência, elas são utilizadas por um modelo de classificação para atribuir um entre muitos rótulos para a entrada em análise. Foram realizados diversos experimentos com o intuito de encontrar o melhor classificador e a melhor técnica de regularização. O classificador com os melhores resultados foi o *Random Forest*, em comparação com seus concorrentes *Softmax, Support Vector Machine* e *k-Nearest Neighbors*. Segundo os autores, as técnicas convencionais de regularização (L1 e L2) possuem desvantagens que precisam ser tratadas para garantir a resolução do problema de *overfitting*. Assim, foi proposto uma nova técnica chamada SD-Reg para tratar eficientemente com o problema. Além disso, foram conduzidos experimentos com subconjuntos reduzidos de características dos dados. Essa redução diminui o custo computacional do sistema e se feita corretamente não diminui consideravelmente o seu desempenho. O modelo resultante dos experimentos CNN-Random-Forest com SD-Reg foi comparado com o modelo recorrente LSTM e obteve resultados superiores nas métricas consideradas. Os dados utilizados nos experimentos são provenientes das bases de dados InSDN, CSE-CIC-IDS2018 e UNSW-NB15. Para medir o desempenho em detecção de anomalias ao longo dos testes foram utilizadas métricas como *Confusion Matrix, Accuracy, Precision, Recall, F1-score* e *ROC curve*.

Rao *et al.* [24] desenvolveram um NIDS baseado em *Sparse Autoencoder (Sparse AE)* e *Deep Neural Network* (DNN) que atua na proteção de redes tradicionais. O sistema trabalha classificando os registros de dados entre normal ou um entre vários rótulos referentes a ataques de redes. Inicialmente, a rede *Sparse AE* é utilizada para reduzir a dimensionalidade da entrada gerando características (*features*) mais significativas do exemplo de dados. Em seguida, a entrada codificada pelo *Autoencoder* é passada para a rede DNN para poder ser classificada. A rede *Sparse AE* aprende uma melhor representação dos dados de entrada, reduzindo o tempo de treinamento e teste da rede DNN e aumentando o sua acurácia de classificação. O modelo Sparse-AE-DNN proposto obteve uma Acurácia de 99,03%, 99,71% e 99,98% nas bases de dados KDDCup99, NSL-KDD e UNSW-NB15, respectivamente. Em comparação com os resultados de outros modelos para essas mesmas bases, como LSTM, SAE-SVM e SAVAER-DNN, a proposta dos autores obteve resultados superiores.

Derhab *et al.* [119] desenvolveram cinco princípios para a construção de um NIDS baseado em Aprendizado Profundo aplicado a ambientes de IoT (*Internet of Things*). Segundo os autores, o sistema precisa: tratar *overfitting*, balancear o conjunto de dados,

efetuar adequadamente a engenharia de características dos dados (*feature engineering*), otimizar a função de custo do modelo de aprendizado de máquina, e utilizar dados que refletem a realidade do ambiente o qual ele será instalado. Seguindo todos esses princípios, os autores propuseram um NIDS leve e eficiente implementado com *Temporal Convolutional Network* (TCN) para ambientes IoT. O sistema recebe como entrada um registro de tráfego e o classifica normal ou um entre quatro tipos de ataques considerados. A base de dados utilizada nos experimentos foi a Bot-IoT, na qual o modelo dos autores obteve uma Acurácia de 99.9986%. O modelo TCN implementado foi comparado com outros como LSTM, CNN, RNN-BPTT e *DeepDCA* e mostrou resultados superiores na tarefa de multi-classificação.

Assis *et al.* [5] propuseram um sistema de defesa contra intrusões e ataques DDoS para ambientes SDN que atua analisando e classificando fluxos IP de maneira individual. Ele é instalado diretamente no controlador SDN e é composto por dois módulos, o de detecção de anomalias e o de mitigação de anomalias. O primeiro foi implementado utilizando uma rede GRU e é responsável por classificar um fluxo IP entre benigno ou maligno. Quando um fluxo é classificado como maligno, um alerta é enviado aos administradores de redes e o módulo de mitigação é acionado. Assim, o segundo módulo trabalha definindo contramedidas para conter o ataque detectado. Além disso, esse módulo também solicita ao controlador a propagação dessas políticas de defesa ao plano de dados da rede. Foram utilizadas duas bases de dados públicas durante o desenvolvimento do trabalho, a CICDDoS2019 e a CICIDS2018. Para mensurar o desempenho do sistema e comparar com outras propostas foram utilizadas métricas como *Accuracy, Precision, Recall, F1-score*, a média da quantidade de fluxos analisados por segundo, a quantidade de fluxos benignos bloqueados e a quantidade de fluxos malignos não bloqueados. A proposta dos autores foi comparado com outros modelos como DNN, CNN, LSTM, *Support Vector Machine*, *Logistic Regression, k-Nearest Neighbors*, e *Gradient Descent*. O modelo GRU implementado apresentou um melhor balanço entre as métricas consideradas.

A Rede Adversária Generativa é um modelo de Aprendizado Profundo que tem sido amplamente estudado e utilizado para resolver os mais diversos problemas [101]. Apesar de seu uso majoritário na área de processamento de imagens, as redes GAN também são objeto de estudo no campo de redes de computadores e de comunicação [32]. Mais especificamente, foi observado um aumento nas pesquisas sobre a aplicação de GAN na área de detecção de anomalias e intrusões de redes [52]. Abaixo estão destacados alguns trabalhos recentes de autores que utilizaram rede GAN na implementação de seus sistemas de detecção de intrusão.

Seo *et al.* [120] construíram um Sistema de Detecção de Intrusão aplicado na proteção de redes de comunicação de dados instaladas em veículos. Essas redes podem sofrer ataques externos que comprometem a integridade física dos motoristas. O sistema

foi implementado utilizando uma combinação de duas redes discriminadoras. A primeira é treinada para agir como um classificador convencional, identificando diferentes ataques cuja assinatura é conhecida. Já a segunda é treinada de maneira adversária juntamente com uma rede geradora para aprender a discernir entre um exemplo de dados real e um exemplo gerado. Deste modo, após o treinamento, o sistema trabalha combinando as capacidades dessas duas redes. A primeira rede discriminadora atua identificando ataques conhecidos e a segunda detectando ataques desconhecidos. Os autores destacam que usar essas duas redes em combinação ao invés apenas da segunda melhora o desempenho geral do sistema. Para a implementação das redes discriminadoras e geradora foi empregado a *Convolutional Neural Network*. Os dados utilizados no desenvolvimento foram coletados pelos próprios autores de um carro YF Sonata da Hyundai e convertidos para formato de imagem. O sistema apresentou uma Acurácia média de 100% para a primeira rede discriminadora e 98% para a segunda. Outras métricas consideradas na avaliação do NIDS incluem *Detection Rate, Precision* e AUC.

Andresini *et al.* [121] desenvolveram um procedimento de construção de Sistemas de Detecção de Intrusão de Redes que trabalha com fluxos IP codificados em formato de imagem, o chamado MAGNETO. Essa codificação possibilita que o sistema seja implementado com modelos de Aprendizado Profundo que são comprovadamente eficazes para trabalhar com esse tipo de entrada, como, por exemplo, CNN e GAN. MAGNETO é composto de três etapas: codificação de dados, aumento de dados e discriminação de dados. A primeira refere-se a transformação das instâncias de dados, os fluxos IP, em formato de imagem em escala de cinza. Na segunda etapa, uma rede GAN é treinada para aprender a distribuição de ataques de rede escassos. Assim, o gerador é utilizado para gerar exemplos sintéticos desses ataques e balancear o conjunto de dados. Os autores destacam a importância de trabalhar com bases de dados balanceadas no desenvolvimento de modelos de Aprendizado de Máquina. Por fim, a etapa três treina uma rede CNN bidimensional para performar a discriminação entre fluxos IP malignos e benignos. Para avaliar a qualidade do sistema construído através de sua metodologia, os cientistas utilizaram quatro bases de dados: KDDCUP99, UNSW-NB15, CICIDS17 e AAGM17. Como métricas de desempenho foram utilizadas a Acurácia e o *F1-score*. O modelo dos autores foi comparado com outros do estado da arte, como aqueles baseados em 1D-CNN, 2D-CNN, LSTM, DNN e GAN. Com base nas métricas consideradas, MAGNETO mostrou-se superior aos concorrentes.

Araujo-Filho *et al.* [122] construíram um software utilizando Rede Adversária Generativa (FID-GAN) para detectar ataques lançados a Sistemas Ciber-Físicos em ambientes *Fog*. A detecção de ataques é modelada como um problema de classificação binária onde se decide se uma instância de dados de rede representa um ataque ou não. O sistema de detecção trabalha calculando pontuações anômalas para cada exemplo de dados em análise, sendo posteriormente utilizadas para classificá-los entre normal ou malicioso.

Essas pontuações são calculadas como uma soma ponderada entre o resultado de classificação do discriminador e o erro de reconstrução do exemplo. Esse erro é obtido ao inserir uma amostra de dados em um *Autoencoder*, onde o decodificador é o próprio gerador da rede GAN, e calcular o *root-mean-square error*. Segundo os autores, considerar o erro de reconstrução da amostra de dados auxilia na tarefa de classificação. Para avaliar o desempenho do sistema desenvolvido, os cientistas utilizaram as bases de dados SWaT, WADI e NSL-KDD. A área sob a curva ROC foi utilizada como métrica e o sistema proposto foi comparado com outros dois similares, o MAD-GAN e o ALAD. Segundo as análises, o modelo dos autores apresentou valores maiores que de seus concorrentes para a métrica considerada em cada um dos conjuntos de dados citados.

O sistema proposto por Nie *et al.* [123] atua na proteção de ambientes *Social Internet of Things* e é composto por três fases. A primeira é responsável por coletar e pré-processar os dados brutos de tráfego. Para realizar o pré-processamento, os autores utilizaram a ferramenta *CICFlowMeter* para extrair *features* dos dados originais, seguido pela aplicação de um algoritmo de *feature selection*. Na segunda fase, para cada tipo distinto de ataque, é realizado o treinamento de um sistema de detecção baseado em rede GAN. Esses sistemas utilizam o seu discriminador aprendido para identificar o ataque em questão. A terceira e última fase agrupa os múltiplos sistemas da fase anterior. Assim, o detector de anomalias final pode reconhecer cada um dos tipos de ataques considerados na fase dois. Para realizar a validação da proposta, os autores testaram o seu desempenho nas bases de dados públicas CSE-CIC-IDS2018 e CIC-DDoS2019, obtendo um *F1-score* geral de 97,30% e 99,17%, respectivamente. Esses resultados mostraram-se superiores aos de concorrentes como *Vector Convolutional Deep Learning, Stacked Non-symmetric Deep Autoencoders, Self-Taught Learning,* e *Stacked Contractive Autoencoder with SVM*.

Zhang *et al.* [124] propuseram um sistema de detecção de *ransomware* baseado em redes GAN chamado de TGAN-IDS (*Transferred Generating Adversarial Network-Intrusion Detection System*). Inicialmente, é executado um pré-treinamento das redes geradora e discriminadora de maneira separada. A rede geradora é treinada adversariamente a partir de uma *Deep Convolutional* GAN. Ao fim do aprendizado, a rede geradora é armazenada e a discriminadora é descartada. Na sequência, uma nova rede discriminadora é treinada de maneira isolada para aprender a distinguir dados normais de dados de ataques conhecidos. Após o término do pré-treino é utilizado a técnica de *Transfer Learning* para inicializar uma nova rede GAN, chamada de TGAN. Assim, inicia-se um novo treinamento adversário a partir das duas redes previamente treinadas. Ao final do processo de otimização, o gerador é descartado e o discriminador é utilizado como sistema de detecção de *ramsomware*. Esse sistema consegue detectar tanto ataques conhecidos quanto desconhecidos. Um ponto importante destacado pelos autores é a introdução de uma função de perda de reconstrução na função de perda original da TGAN. Essa alteração garante uma melhor convergência durante o treinamento. As bases de dados utilizadas no

desenvolvimento incluem *Malware-Traffic-Analysis.net, PacketTotal, Interactive Online Malware Analysis Sandbox*, CICIDS2017, KDD99, SWaT e WADI. Trabalhos similares como ALAD, MAD-GAN e FID-GAN foram comparados com a proposta dos autores, que obteve um desempenho geral superior nas métricas consideradas.

Novaes *et al.*, [31], Novaes *et al.* [28] e Lent *et al.* [7] desenvolveram um NIDS baseado em Aprendizado Profundo para proteger redes SDN. Diferentemente dos trabalhos citados anteriormente, essas propostas realizam a análise de tráfego em intervalos de um segundo ao invés de performar a análise convencional fluxo a fluxo. Assim, o sistema trabalha coletando e processando todos os fluxos IP observados no último segundo, produzindo um registro de dados que representa o resumo do tráfego neste segundo. Na sequência, essa observação é analisada pelo módulo de detecção de anomalias, que invoca o módulo de mitigação caso uma anomalia seja identificada. O NIDS executa no plano de aplicação da rede SDN e se comunica com o controlador SDN para requisitar dados de tráfego a todo segundo e para enviar as políticas de mitigação definidas. Em [31] o módulo de detecção de anomalias foi implementado como um modelo de classificação binária que classifica os registros de dados como normal ou anômalo. Ele foi implementado com base em GAN e aprendizado adversário, tornando-o menos sensível aos chamados ataques adversários, uma vulnerabilidade de modelos de Aprendizado de Máquina. Em [28] os autores incluíram um módulo extra que atua antes do módulo que detecção para realizar a caracterização do tráfego. Esse módulo foi implementado com LSTM e trabalha realizando a previsão do comportamento esperado do tráfego. Na sequência, com base em Lógica *Fuzzy*, o módulo de detecção de anomalias compara o comportamento previsto com o comportamento real observado do tráfego. Caso a diferença entre os dois ultrapasse um certo limiar, o módulo indica a presença de anomalia nos dados. O trabalho [7] tem uma implementação semelhante ao [28] e o seu módulo de caracterização é implementado utilizando GRU. Para validar a sua proposta, os três trabalhos utilizaram dois conjuntos de dados, um gerado e disponibilizado pelo grupo de pesquisa ORION da UEL e o CICDDoS2019. O desempenho de seu sistema sobre essas bases de dados foi avaliada com o cálculo de métricas como *Accuracy, Precision, Recall, F1-score, ROC curve* e comparada com a de outras propostas.

É apresentado na tabela 2 um resumo sobre os trabalhos relacionados descritos neste capítulo. A partir da revisão da literatura pode-se concluir que, de fato, a detecção de anomalias e intrusões em redes é um campo de pesquisa que demanda interesse substancial da comunidade científica. Além disso, muitos trabalhos recentes têm proposto soluções que exploram o potencial da Rede Adversária Generativa. Assim, este trabalho planeja estudar a aplicabilidade da rede GAN na implementação de um Sistema de Detecção de Intrusão de Redes para auxiliar na proteção de redes SDN.

Tabela 2 – Trabalhos relacionados.

Trabalho	**Ano**	**Modelo(s)**	**Ambiente**	**Conjunto de dados**
Vinayakumar *et al.* [19]	2019	DNN	Tradicional	KDDCup99, NSL-KDD, UNSW-NB15, Kyoto, WSN-DS e CICIDS2017
Khan *et al.* [21]	2019	DSAE	Tradicional	KDDCup99 e UNSW-NB15
Seo *et al.* [120]	2019	CNN + GAN	*In-vehicle network*	Próprio
Derhab *et al.* [119]	2020	TCN	IoT	BoT-IoT
Assis *et al.* [5]	2020	GRU	SDN	CICDDoS2019 e CICIDS2018
Novaes *et al.* [28]	2020	LSTM	SDN	Próprio e CICDDoS2019
Araujo-Filho *et al.* [122]	2020	GAN + *Autoencoder*	*Fog*	SWaT, WADI e NSL-KDD
ElSayed *et al.* [27]	2021	CNN + *Random Forest*	SDN	InSDN, CSE-CIC-IDS2018 e UNSW-NB15
Rao *et al.* [24]	2021	SAE + DNN	Tradicional	KDDCup99, NSL-KDD e UNSW-NB15
Novaes *et al.* [31]	2021	GAN	SDN	Próprio e CICDDoS2019
Nie *et al.* [123]	2021	GAN	*Social IoT*	CSE-CIC-IDS2018 e CIC-DDoS2019
Zhang *et al.* [124]	2021	DCGAN	Tradicional	*Malware-Traffic-Analysis.net, PacketTotal, Interactive Online Malware Analysis Sandbox*, CICIDS2017, KDDCup99, SWaT e WADI
Andresini *et al.* [121]	2021	GAN + CNN	Tradicional	KDDCup99, UNSW-NB15, CICIDS17 e AAGM17
Lent *et al.* [7]	2022	GRU	SDN	Orion e CICDDoS2019

5 ESTUDO DE CASO

A seguir será apresentado um estudo que avalia a aplicabilidade de Redes Adversárias Generativas na implementação de um NIDS. Inicialmente, tem-se a especificação do ambiente de redes em que espera-se instalar e executar esse sistema. Além disso, descreve-se a base de dados utilizada para simular esse ambiente durante o desenvolvimento do NIDS. As seções subsequentes introduzem o Sistema de Detecção de Intrusão de Redes que foi estudado e desenvolvido. Um segundo sistema baseado em rede GRU também é implementado. Os desempenhos de ambos os sistemas no ambiente de rede simulado são avaliados e comparados entre si com base em métricas quantitativas.

5.1 Ambiente de Execução

O NIDS desenvolvido têm como propósito promover segurança em Redes Definidas por Software. É ilustrada na Figura 15 uma visão global do ambiente de redes considerado. O sistema de detecção é instalado no plano de aplicação da rede SDN. Ele se comunica com o controlador SDN através da interface *northbound* para coletar dados de tráfego e realizar as análises.

5.1.1 Fluxo IP

O tipo de dados utilizado neste estudo de caso para representar o tráfego de rede é o fluxo IP. Ele pode ser definido como um conjunto de pacotes IP que passaram por um ponto específico da rede durante um certo intervalo de tempo [125]. Os pacotes que constituem um fluxo IP possuem propriedades em comum, como, por exemplo, endereço de IP de origem, endereço de IP de destino, porta de origem, porta de destino e protocolo [15]. Um fluxo IP fornece informações como: o tempo de início e duração da troca de dados, os IPs e portas que participaram da comunicação e o total de bytes e pacotes transferidos. Muitos trabalhos da área de gerência de redes utilizam o formato de fluxo IP para representar e analisar o tráfego [126], [127], [128]. Processar pacotes individualmente (*deep packet inspection*) tornou-se extremamente custoso devido às altas velocidades de troca de dados alcançadas nas redes atuais [129].

5.1.2 Conjunto de Dados

Durante o desenvolvimento de um NIDS é comum o uso de conjuntos de dados de tráfego publicamente disponíveis [130], [131]. Esses conjuntos são compostos por fluxos IP que representam longos períodos de tráfego de rede. Os desenvolvedores simulam a coleta de dados periódica feita em ambientes reais a partir da extração constante de partes desses

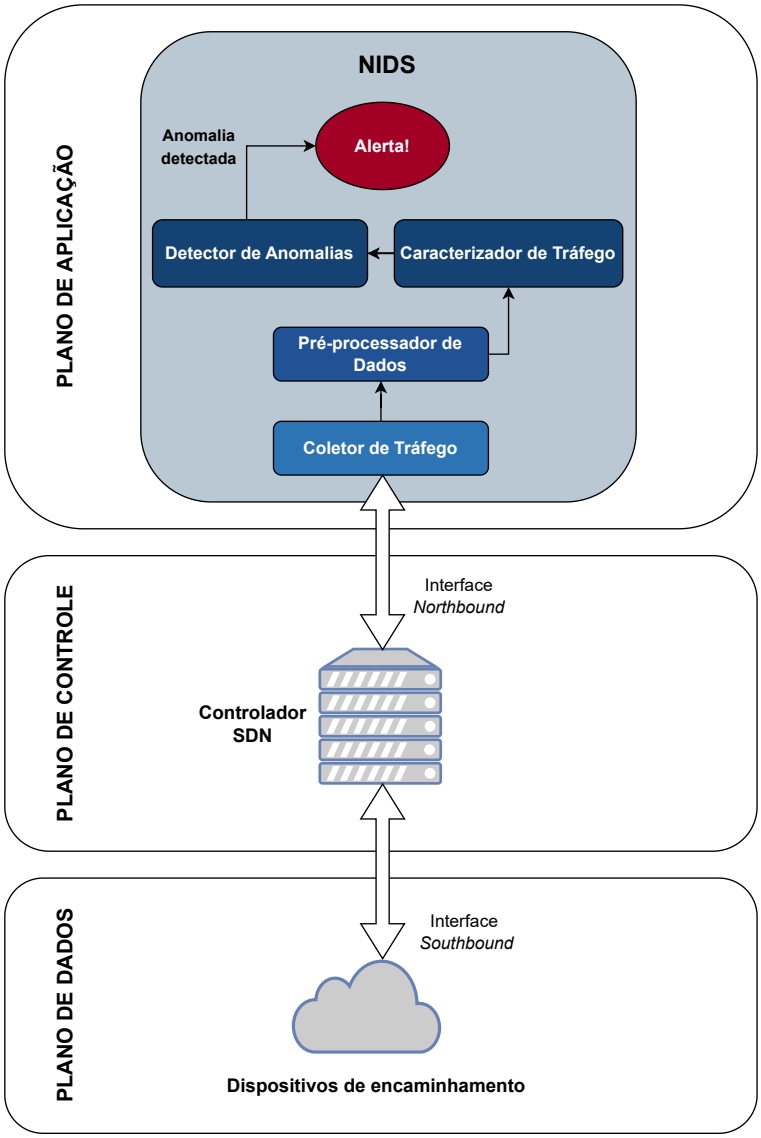

Figura 15 – Ambiente de redes considerado.

conjuntos. Dados públicos também são úteis para comparar a capacidade de detecção de anomalias de diferentes implementações de NIDS, pois, fornecem um ambiente comum para o teste de propostas distintas.

O grupo de pesquisa ORION da Universidade Estadual de Londrina gerou uma base de dados no formato de fluxos IP para auxiliar no desenvolvimento dos seus sistemas de segurança [8]. Os pesquisadores utilizaram a ferramenta *Mininet* para emular uma Rede Definida por Software. O plano de dados dessa rede é formado por seis *switches* organizados em uma topologia de árvore, como ilustrado na Figura 16. Cada um dos cinco *switches* filhos são conectados a doze *hosts*, totalizando sessenta computadores na rede. O tráfego de rede foi emulado utilizando a biblioteca *Scapy*. Ataques DDoS e *Porstcan* direcionados a essa rede foram simulados utilizando o software *hping3* e a biblioteca *Scapy*,

respectivamente.

A base de dados ORION é formada por três dias inteiros de tráfego de rede, está completamente rotulada e encontra-se publicamente disponível em [132]. O primeiro dia de dados corresponde a um dia de funcionamento normal da rede. Nesse dia a rede não sofreu nenhum dos dois ataques mencionados anteriormente. O segundo e terceiro dia contém tráfego anômalo em diferentes intervalos provenientes de ataques DDoS e *Portscan*. Essa base de dados foi utilizada neste trabalho para simular um ambiente de rede SDN. O estudo de caso desenvolvido foca em detectar, especificamente, ataques DDoS. Esse tipo de ataque é estudado continuamente pela comunidade científica, sendo considerado um dos mais destrutivos à rede [133], [134].

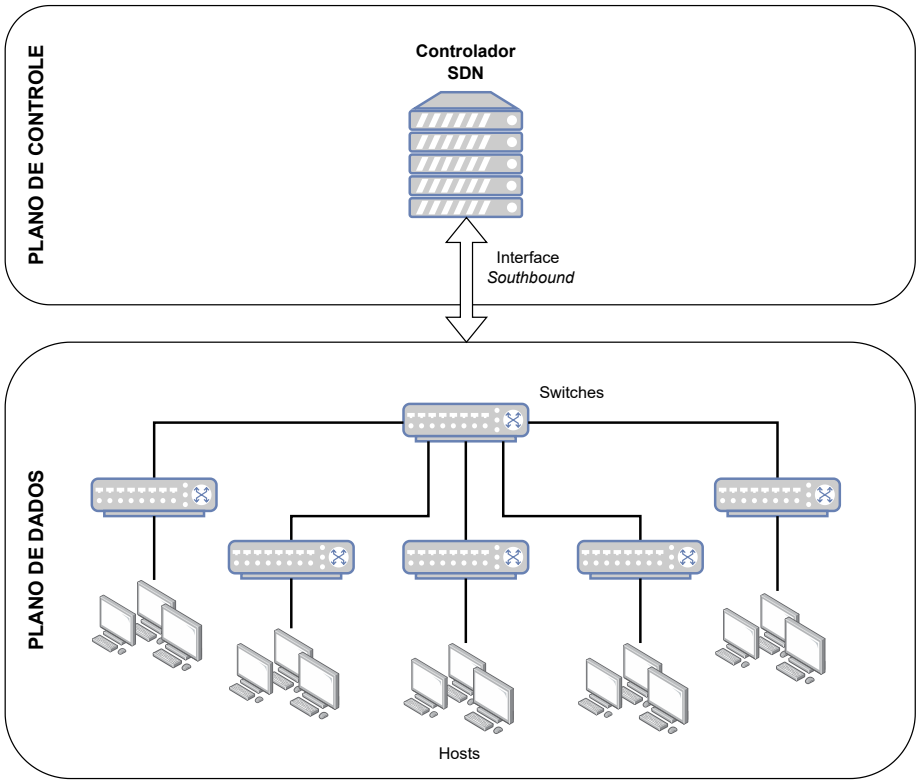

Figura 16 – Simulação de um ambiente de redes.

5.2 Sistema de Detecção de Intrusão de Redes

O Sistema de Detecção de Intrusão de Redes estudado é composto por quatro módulos que realizam as seguintes funções de maneira ordenada, como ilustrado na Figura 15: coleta de dados, pré-processamento de dados, caracterização de tráfego e detecção de anomalias. Diversos trabalhos do grupo de pesquisa ORION utilizaram uma metodologia semelhante de implementação e obtiveram reconhecimento da comunidade científica internacional [7], [31], [5], [28].

5.2.1 Coleta de Dados

Um NIDS detecta anomalias e ataques de rede através da análise dos dados do tráfego. Portanto, é necessário que esses dados sejam coletados constantemente para o sistema poder avaliá-los. Um dos serviços que o controlador de uma rede SDN fornece para o plano de aplicação é a exportação de dados de tráfego. Para usufruir desse serviço, o sistema de detecção se comunica com o controlador através da interface *northbound* utilizando um módulo dedicado, o Coletor de Tráfego.

O intervalo de coleta e análise é uma variável cujo valor reduziu-se ao longo do tempo devido à alta taxa de transmissão alcançada em ambientes de redes atualmente [31]. É importante que os períodos entre os momentos de análise sejam pequenos, possibilitando uma rápida detecção e mitigação de possíveis ataques. Por exemplo, os sistemas propostos por [8] e [29] efetuam a coleta de dados a cada segundo.

O NIDS deste estudo de caso trabalha coletando e analisando o tráfego de rede em intervalos de um segundo. Assim, a cada segundo o seu módulo de coleta de tráfego requisita ao controlador SDN os fluxos IP que representam o tráfego de rede no último segundo.

5.2.2 Pré-processamento de Dados

Para cada instante de análise de tráfego, os fluxos IP coletados passam por uma etapa de pré-processamento antes de serem avaliados pelo NIDS. Essa transformação tem como entrada o conjunto de fluxos IP referente ao último segundo de tráfego, e produz como saída uma observação estatística. Essa observação provê um resumo sobre o tráfego naquele segundo. Ela é composta por seis atributos: total de bits transferidos, total de pacotes transferidos, entropia de IP de origem, entropia de IP de destino, entropia de porta de origem, entropia de porta de destino. Essas características são calculadas considerando as informações presentes em cada um dos fluxos IP coletados.

A entropia utilizada é a Entropia de Shannon [135]. Ela representa o cálculo que converte um conjunto de valores qualitativos em um único valor quantitativo, que indica a medida de dispersão ou concentração desses dados [29]. Essa conversão é necessária, pois modelos de Aprendizado de Máquina processam apenas valores quantitativos [7]. A entropia de um conjunto de valores qualitativos aumenta quando este possui uma distribuição relativamente uniforme entre eles, pois os dados estão dispersos. Por outro lado, quando certos valores possuem mais ocorrências do que os demais, a entropia do conjunto diminui devido à concentração dos dados.

Por exemplo, durante um ataque DDoS, um *host* específico é sobrecarregado com múltiplas requisições de diversos computadores atacantes, causando mudanças súbitas nos valores de entropia. As entropias de IP e porta de destino são reduzidas, uma vez que

o endereço da vítima é observado com mais frequência durante o ataque. Já a entropia de porta de origem aumenta, pois os agentes maliciosos enviam as requisições utilizando portas aleatórias, ampliando a dispersão desses dados [28], [59]. Um ataque *Portscan* objetiva escanear as portas de um *host* alvo para identificar potenciais vulnerabilidades e permitir que outros ataques sejam lançados. Durante a ocorrência desse tipo de ataque, a entropia de porta de destino tende a aumentar, pois, o agente malicioso envia requisições para múltiplas portas da vítima, causando o espalhamento dos dados [6].

A equação 5.1 descreve o cálculo da Entropia de Shannon. A variável x_t^A representa o histograma da quantidade de ocorrências dos diferentes possíveis valores do atributo qualitativo A no tempo de análise t. Tem-se que $x_t^A = \{x_1, x_2, ..., x_N\}$, onde x_i indica a quantidade de ocorrências do valor i. S é definido por $\sum_{i=1}^{N} x_i$, isto é, o total de valores observados para o atributo A no tempo de análise.

$$H(x_t^A) = -\sum_{i=1}^{N}(\frac{x_i}{S})log_2(\frac{x_i}{S}) \tag{5.1}$$

São apresentadas na Figura 17 as seis características descritas anteriormente calculadas para um dia inteiro de tráfego (86400 segundos). Esse tráfego corresponde ao primeiro dia da base ORION. Pode-se notar uma variação suave nas seis características ao longo do dia devido à ausência de ataques de rede. Também é ilustrado na Figura 18 e Figura 19 o resultado do pré-processamento para o segundo e terceiro dia de tráfego, respectivamente. Estão indicados em vermelho os intervalos do dia onde os ataques DDoS são lançados contra a rede emulada. No segundo dia os ataques DDoS são executados no período da manhã, entre 10:15:00 e 11:30:00. No terceiro dia os ataques ocorrem entre o fim da tarde e o início da noite, iniciando às 17:37:00 e finalizando às 18:55:00. Notam-se bruscas mudanças nas características de tráfego para esses intervalos devido à presença dos ataques.

Após o cálculo da observação estatística que resume o tráfego no segundo em análise, tem-se a segunda e última etapa de pré-processamento: o *scaling*. Diferentes atributos de uma observação de dados podem estar dispostos em intervalos numéricos diferentes, como pode-se notar na Figura 17. Essa diferença de intervalos prejudica o treinamento de algoritmos de Aprendizado de Máquina [136]. O *scaling* reduz e padroniza os intervalos das características. Existem diversos métodos para aplicá-lo nos dados, como normalização e padronização [137]. Por exemplo, o método de normalização *MinMax* converte proporcionalmente um conjunto de valores para um intervalo específico como [0, 1] ou [-1, 1]. O módulo seguinte do sistema é implementado utilizando Aprendizado de Máquina, logo, realiza-se o *scaling* na observação de tráfego calculada.

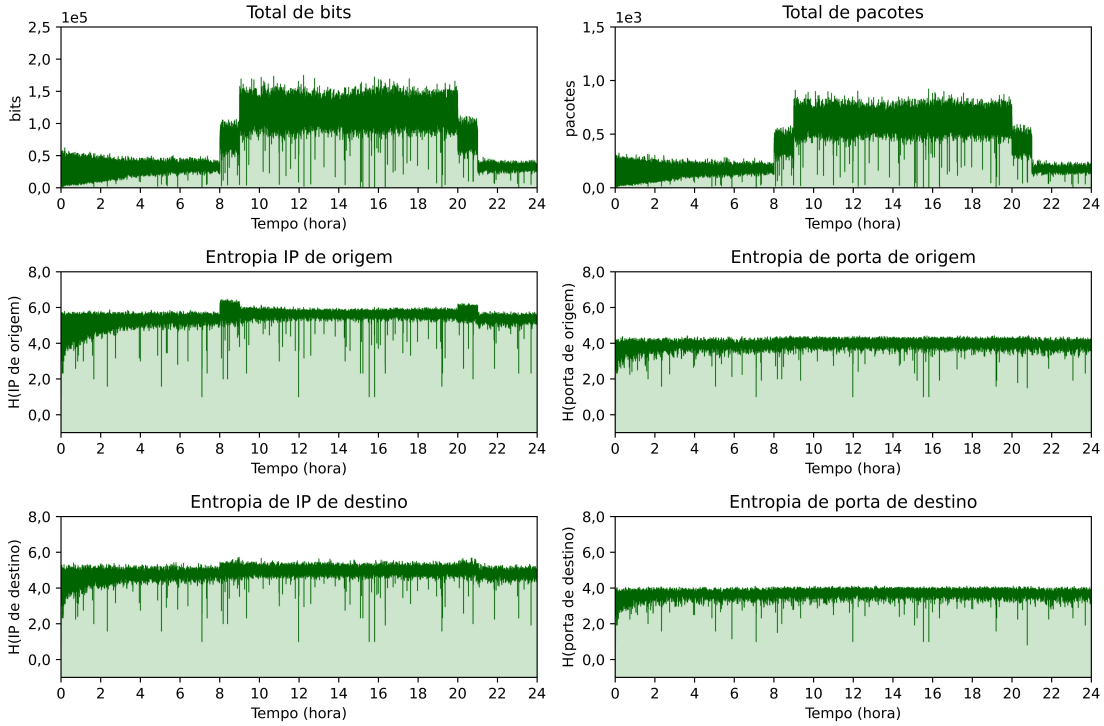

Figura 17 – Dia 1 ORION (pré-processamento sem *scaling*)

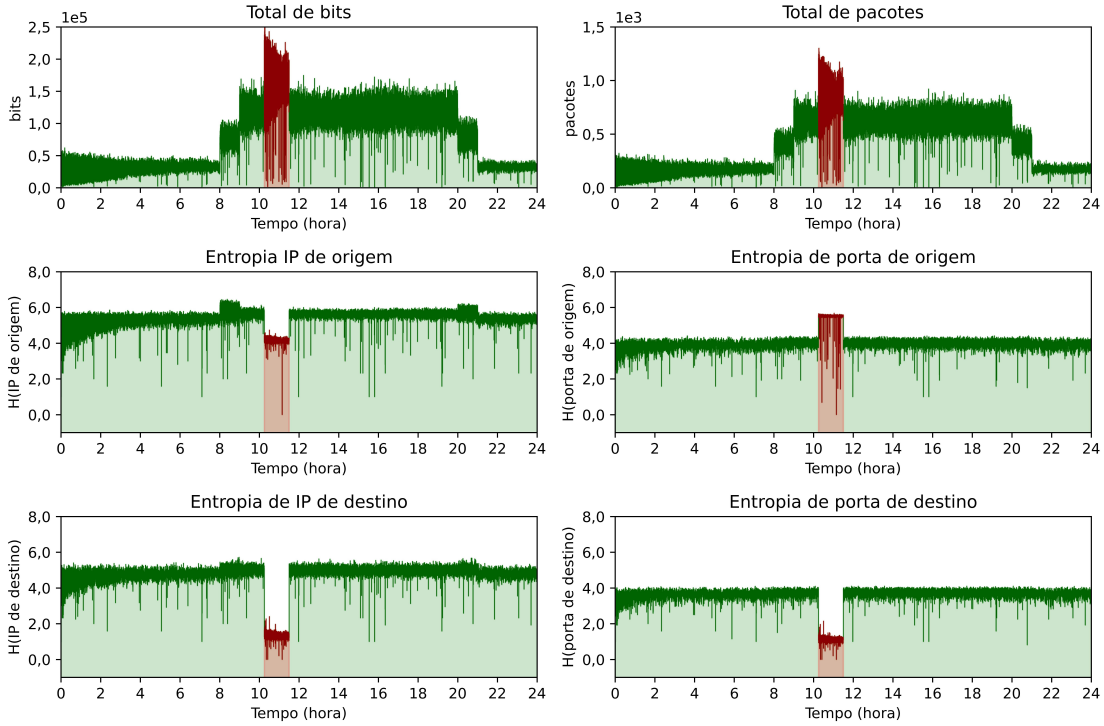

Figura 18 – Dia 2 ORION (pré-processamento sem *scaling*)

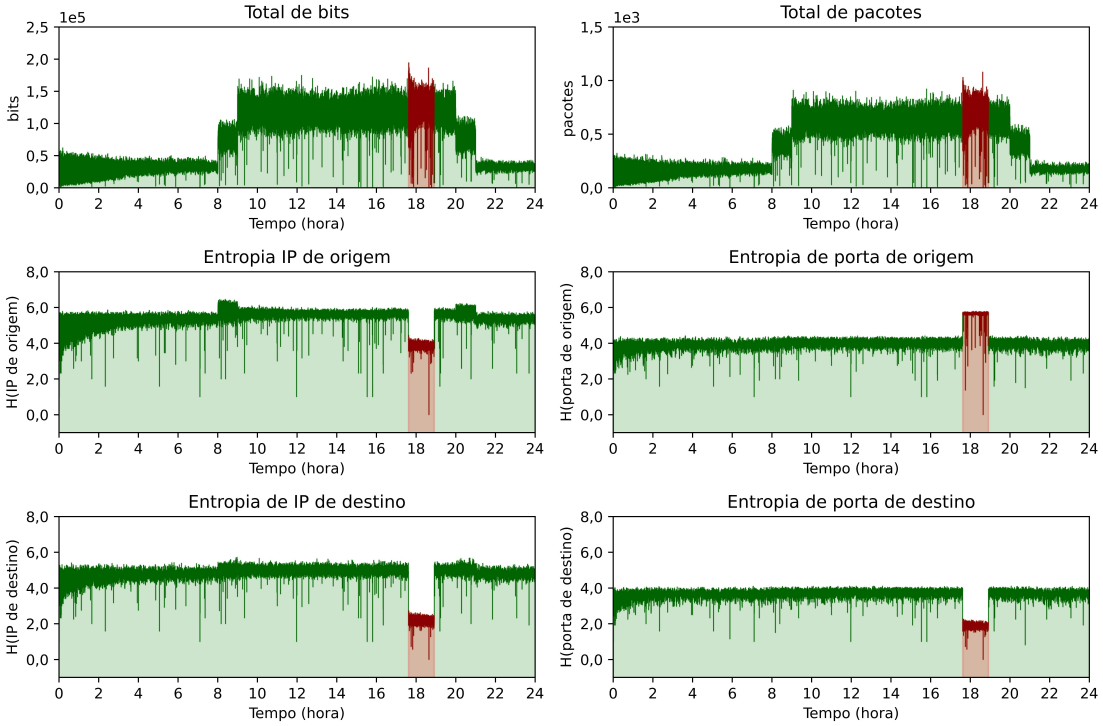

Figura 19 – Dia 3 ORION (pré-processamento sem *scaling*)

5.2.3 Caracterização de Tráfego

Na etapa de desenvolvimento do NIDS, o módulo de caracterização de tráfego define o padrão de comportamento normal do tráfego de rede (*baseline*). Assim, antes do NIDS ser instalado para proteger a rede, o seu módulo de caracterização é alimentado com dados históricos do tráfego para que este possa aprender a mapear o comportamento da rede. Posteriormente, esse aprendizado será utilizado na detecção de anomalias quando o sistema for introduzido na rede. Foi usado o modelo de Rede Adversária Generativa para implementar o módulo de caracterização. Um segundo módulo baseado em Rede Neural Recorrente GRU também foi implementado para fins de comparação, como será descrito nas próximas seções.

5.2.4 Detecção de Anomalias e Geração de Alerta

Para cada instante de análise, o módulo de detecção de anomalias performa a tarefa de comparar o padrão de comportamento apresentado pela observação estatística com o *baseline* aprendido pelo módulo anterior na etapa de desenvolvimento do NIDS. Se a diferença entre eles ultrapassar um certo limiar (*threshold*) pré-definido, esse subsistema alertará os administradores de rede da presença de anomalia nos dados. A ocorrência de anomalias causa mudanças incomuns nas características de tráfego. Como consequência, o padrão comportamental da rede neste momento difere do perfil normal aprendido pelo

módulo de caracterização de tráfego, e a sua diferença possivelmente ultrapassa o limiar. Assim como a modelagem do *baseline* pelo módulo de caracterização de tráfego, o limiar de decisão também é definido antes do NIDS ser instalado na rede.

5.3 Sistemas Desenvolvidos

Foram elaboradas duas versões do sistema de segurança descrito na seção anterior, como ilustrado na Figura 20. A primeira foi implementada com base em Redes Adversárias Generativas (NIDS-GAN). A segunda versão foi construída utilizando Rede Neural Recorrente GRU (NIDS-GRU), que foi escolhida devido a sua adequabilidade no processamento de séries temporais [7], como dados de tráfego de rede.

O primeiro módulo, o de coleta de dados, é o mesmo para ambas as versões. Os seus subsistemas de pré-processamento são semelhantes, diferenciando-se pelo método de *scaling* utilizado e pela formatação dos dados. Os NIDS possuem implementações contrastantes para os módulos de caracterização de tráfego e de detecção de anomalias. Uma descrição detalhada da implementação de cada sistema é apresentada a seguir.

5.3.1 NIDS-GAN

Este sistema utiliza na etapa de pré-processamento o método *MinMaxScaler* para realizar o *scaling* nos dados. Ele é disponibilizado pelo pacote *scikit-learn* [138] da linguagem *Python*. Os dados passam por uma normalização, sendo reduzidos proporcionalmente para estarem dispostos no intervalo [-1, 1]. A escolha desse método de *scaling* e de seu intervalo foi realizada com base nas recomendações para treinamento de redes GAN descritas por Radford *et al.* [139].

5.3.1.1 Caracterização de Tráfego

O módulo de caracterização de tráfego foi implementado a partir do modelo de Rede Adversária Generativa. Durante a etapa de desenvolvimento do NIDS, a rede GAN é treinada utilizando dados históricos de tráfego que não possuem ataques (dia 1 ORION). Mais especificamente, a unidade de dados com a qual a rede trabalha é a observação estatística de tráfego proveniente do módulo de pré-processamento. Assim, o trabalho do gerador é o de sintetizar observações que sejam semelhantes às reais. Enquanto a tarefa do discriminador é a de rotular as observações que realmente são provenientes do tráfego como 0 e as que foram geradas como 1.

Após o treinamento, espera-se que o gerador aprenda uma distribuição que seja próxima da distribuição das observações benignas de tráfego e gere exemplos que se aproximem dela. Devido ao aprendizado do gerador, a rede discriminadora será incapaz de

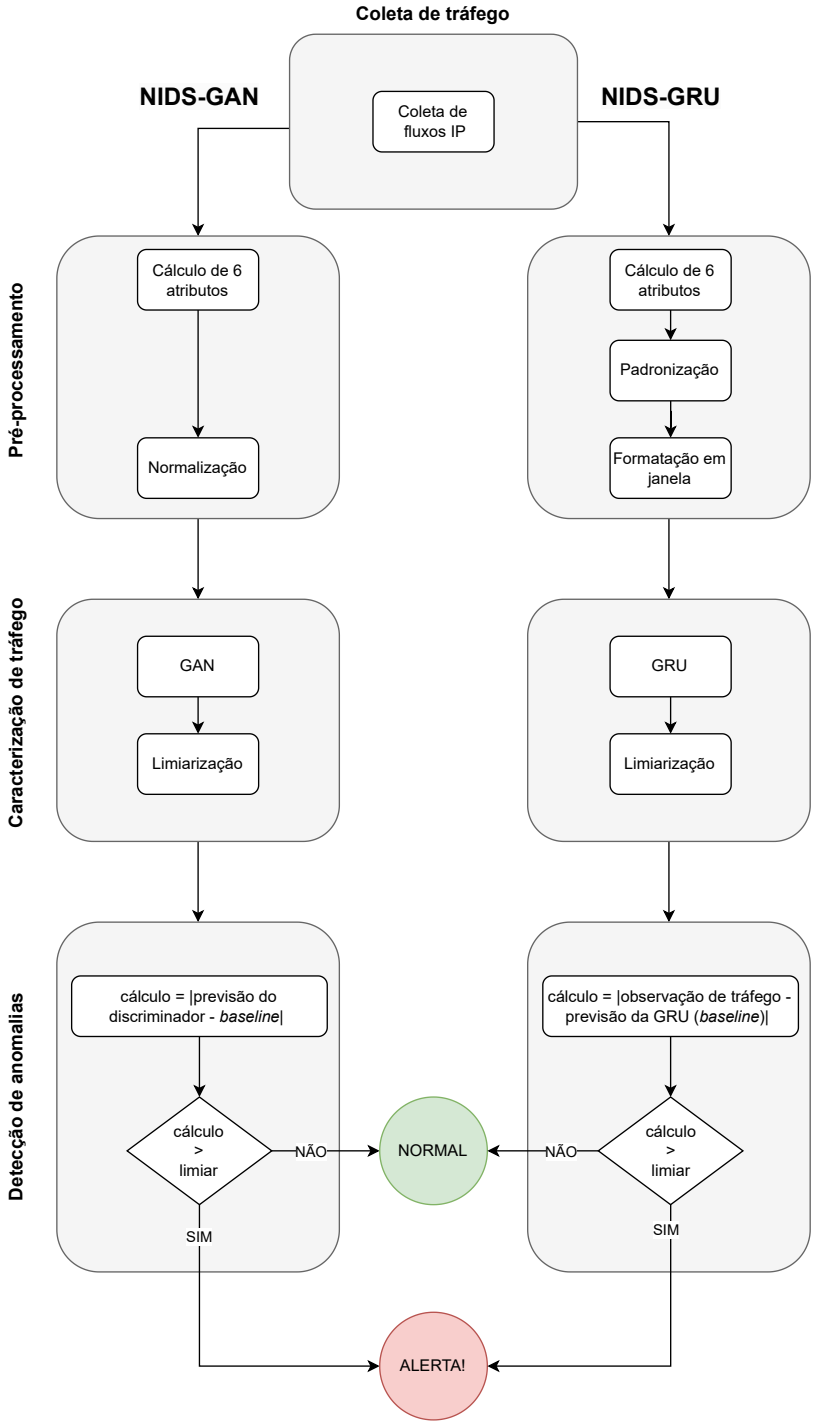

Figura 20 – NIDS-GAN e NIDS-GRU.

decidir se uma observação é sintética ou original, produzindo saídas próximas de 0,5. Esse valor indica a sua incerteza quanto a classificação da entrada analisada.

A rede GAN treinada possui uma representação interna do *baseline* comportamental da rede. É possível utilizar o discriminador para definir um *baseline* externo que possa ser utilizado para detectar anomalias. O NIDS-GAN tem como *baseline* a média

m das saídas produzidas pelo discriminador para observações estatísticas de tráfego do dia 1 da base ORION. Esse *baseline* é estático e não muda independente do momento de análise. É esperado que o valor de m esteja próximo de 0,5, pois todas essas observações se encaixam na distribuição de tráfego benigno e causam incerteza na rede discriminadora durante a sua classificação. Após a etapa de desenvolvimento, quando o sistema é instalado na rede, o módulo de caracterização utiliza o discriminador treinado para analisar observações estatísticas e produzir o seu padrão comportamental (saída da rede). Esse padrão juntamente com o *baseline* definido na etapa de desenvolvimento são utilizados nos cálculos realizados pelo módulo seguinte, o de detecção de anomalias.

5.3.1.2 Limiarização

A rede discriminadora foi treinada apenas com observações de tráfego benignas. Ao efetuar a análise de uma observação anômala, essa rede produz uma saída que se distancia de 0,5. A incerteza da rede diminui, pois essa entrada não se encaixa na distribuição original de dados. Ou seja, o discriminador a classifica como uma observação sintética proveniente do gerador (anômala). Um ponto importante a ser definido é o quanto a saída do discriminador (padrão comportamental da observação) precisa se distanciar do *baseline* (média m) para o exemplo de dados em análise ser considerado anômalo.

Para definir esse limiar, executa-se um algoritmo de busca que utiliza como entrada o discriminador treinado, a média m e um conjunto de dados com observações anômalas (dia 2 ORION). Inicialmente, o algoritmo efetua a análise do conjunto de dados utilizando o discriminador. As diferenças entre as saídas produzidas pela rede e a média m são calculadas. Como já mencionado, espera-se que a saída do discriminador esteja próxima de m ($\approx 0{,}5$) para observações benignas de entrada e distante desse valor para observações anômalas. Após essa etapa, o algoritmo possui as distâncias de cada uma das observações analisadas em relação ao *baseline*. Por fim, realiza-se a busca pela distância máxima (limiar) que uma observação pode ter do *baseline* para ser considerada benigna. O intervalo de busca ideal foi definido empiricamente como $[0, 10^{-1}]$.

A cada iteração de busca seleciona-se um valor dentro desse intervalo, utilizando-o para calcular os rótulos de cada observação da seguinte maneira: caso a distância entre uma observação e o *baseline* seja maior que o limiar ela é considerada anômala, se não ela recebe o rótulo de benigna. Utilizando a rotulação inferida através do limiar em análise e a rotulação real disponibilizada pela base de dados ORION, calcula-se a métrica *Matthews Correlation Coefficient* (MCC). O MCC é comumente utilizado para avaliar modelos de classificação binária [88]. A descrição dessa métrica é apresentada na seção 5.4.1. Ao fim das iterações, armazena-se o limiar que prover a maior pontuação na métrica MCC para o sistema. Esse limiar representa o ponto ótimo de fronteira entre observações benignas e anômalas para o conjunto analisado. Espera-se que o limiar seja genérico e que continue

sendo representativo em futuras análises de dados.

5.3.1.3 Detecção de Anomalias

O módulo de detecção de anomalias é responsável por avaliar o padrão comportamental apresentado pela observação de dados e identificar potenciais traços anômalos. Dada uma observação de tráfego referente ao segundo de análise, esse subsistema inicialmente calcula a distância entre a saída produzida pelo discriminador para essa entrada e a média m. Caso a distância ultrapasse o limiar, a observação é considerada anômala e um alarme é soado para os administradores de rede.

Em suma, o NIDS-GAN apresentado possui a fase de desenvolvimento (treinamento) onde se define o *baseline* comportamental da rede, um mecanismo de extração de comportamento de observações de tráfego (discriminador treinado) e um limiar de tolerância. Em uma segunda fase, o sistema é instalado na rede e utiliza essas definições para analisar dados nunca avaliados. A capacidade de detecção de anomalias do NIDS-GAN foi avaliada considerando um terceiro dia de tráfego não utilizado durante a etapa de desenvolvimento (dia 3 ORION). Os resultados da avaliação são apresentados na Seção 5.4.

5.3.2 NIDS-GRU

O NIDS-GRU utiliza na etapa de pré-processamento o método *StandardScaler* (padronização) para realizar o *scaling* nos dados. Ele é disponibilizado pelo pacote *scikit-learn* [138] da linguagem *Python*. Os dados foram dispostos próximos a uma média contendo um certo desvio padrão. Esse método foi escolhido empiricamente. Outras formas de *scaling* prejudicavam o aprendizado da rede GRU.

Devido a sua natureza, as redes GRU trabalham com janelas de observações de dados. Essa rede recebe uma janela de entrada e produz uma saída. Para atender a esse requerimento, as observações estatísticas de tráfego são agrupadas em janelas ordenadas temporalmente.

5.3.2.1 Caracterização de Tráfego

O módulo de caracterização de tráfego foi implementado a partir do modelo de Rede Neural Recorrente GRU. Durante a etapa de desenvolvimento do NIDS, a rede GRU é treinada utilizando janelas de observações benignas de tráfego (dia 1 ORION). Diferentemente da rede GAN que performa geração de dados e classificação binária de exemplos de dados, a rede GRU realiza a tarefa de previsão de série temporal (regressão). O seu trabalho se resume a mapear uma janela de observações de tráfego para a observação que se localiza imediatamente após essa janela na linha do tempo, como ilustrado na Figura 21.

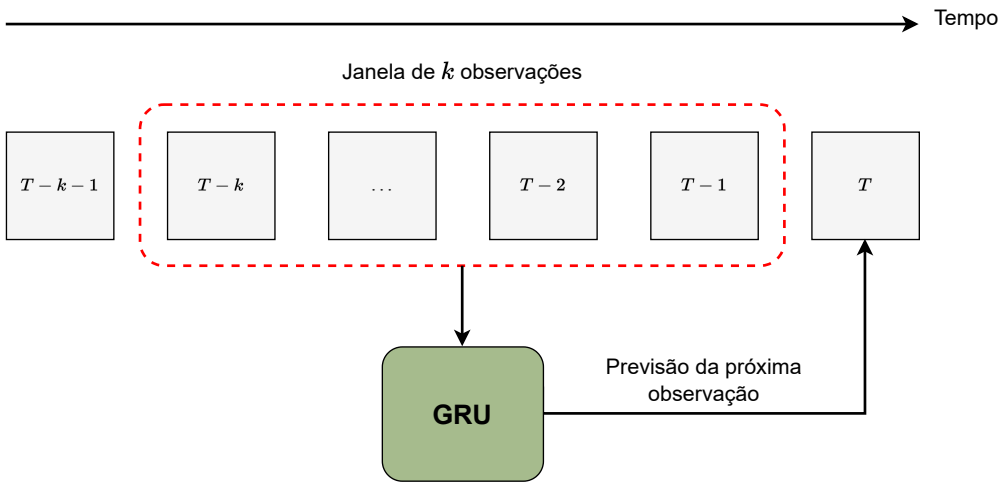

Figura 21 – Previsão de série temporal.

Após o treinamento, a rede GRU aprende a prever como será o comportamento da rede no segundo do dia em análise com base em uma janela de comportamentos observados anteriormente. Espera-se que essa previsão seja relativamente próxima do comportamento real observado no tráfego. Assim, pode-se afirmar que a rede GRU treinada possui uma representação interna do *baseline* de comportamento normal da rede. O NIDS-GRU possui um *baseline* dinâmico (previsão da rede GRU) que muda conforme a janela de comportamentos observada anteriormente. Após a etapa de desenvolvimento, quando o sistema é instalado na rede, o módulo de caracterização usa a rede GRU para gerar o *baseline* para o instante de análise, utilizado pelo módulo seguinte para detectar anomalias.

5.3.2.2 Limiarização

A rede GRU foi treinada para performar previsão de séries temporais de tráfego cujas observações são todas benignas. Devido ao treinamento, a rede consegue realizar a previsão comportamental de uma observação benigna em um certo tempo t com um erro mínimo. Porém, a previsão do sistema possuirá um erro maior quando observação de tráfego em análise for anômala. Observações anômalas possuem um padrão numérico diferente daquele aprendido pela rede GRU. Assim, é necessário definir o quanto o comportamento observado precisa se distanciar do *baseline* (previsão da rede GRU) para a observação em análise ser considerada anômala.

Para definir esse limiar, executa-se um algoritmo de busca semelhante ao da NIDS-GAN. Ele utiliza como entrada a rede GRU treinada e um conjunto de dados com observações anômalas (dia 2 ORION). Inicialmente, o algoritmo efetua a previsão comportamental de cada observação do conjunto de dados utilizando a rede GRU. As diferenças entre as previsões (*baseline*) produzidas pela rede e o comportamento real observado são calculadas. Após essa etapa, o algoritmo possui as distâncias de cada uma das observações

analisadas em relação ao *baseline*. Por fim, realiza-se a busca pela distância máxima (limiar) que uma observação pode ter do *baseline* para ser considerada benigna. O intervalo de busca ideal foi definido empiricamente como $[0, 30]$.

A cada iteração de busca seleciona-se um valor dentro desse intervalo, utilizando-o para calcular os rótulos de cada observação da seguinte maneira: caso a distância entre uma observação e o *baseline* seja maior que o limiar ela é considerada anômala, se não ela recebe o rótulo de benigna. Com base na rotulação inferida através do limiar selecionado e os rótulos reais disponibilizados pela base de dados ORION, calcula-se a métrica *Matthews Correlation Coefficient* (MCC). Ao fim das iterações, o limiar que prover a maior pontuação na métrica MCC para o sistema é registrado para ser usado subsequentemente. Esse limiar representa o ponto ótimo de divisão entre observações benignas e anômalas para o conjunto analisado. Espera-se que essa divisão seja genérica e que continue sendo representativa em futuras análises de dados.

5.3.2.3 Detecção de Anomalias

O módulo de detecção de anomalias responsabiliza-se pela identificação de anomalias nas observações de tráfego. Dada uma observação de tráfego referente ao segundo de análise, esse subsistema calcula a distância entre a previsão comportamental (*baseline*) gerada pela rede GRU e o comportamento real da observação. Caso a distância ultrapasse o limiar, a observação é considerada anômala e um alarme é soado para os administradores de rede.

Em resumo, o NIDS-GRU apresentado possui a fase de desenvolvimento (treinamento) onde se define um mecanismo de geração dinâmico de *baseline* comportamental de tráfego (rede GRU) e um limiar de tolerância. Em uma segunda fase, quando o sistema é instalado para proteger uma rede, essas definições são utilizadas para avaliar o tráfego e identificar anomalias. A capacidade de detecção de anomalias do NIDS-GRU foi avaliada e comparada com a do NIDS-GAN considerando um terceiro dia de tráfego não utilizado durante a sua etapa de desenvolvimento (dia 3 ORION). Os resultados da avaliação são apresentados na Seção 5.4.

5.4 Resultados e Discussão

Ambos os NIDS apresentados foram executados a partir da base de dados ORION especificada na Seção 5.1.2. Esse conjunto de dados simula um ambiente comum para avaliar e comparar o seu desempenho em detecção de anomalias. Os dois primeiros dias de tráfego são utilizados para configurar (treinar) os sistemas. Os resultados dos módulos de caracterização de tráfego são ilustrados. Em seguida, o terceiro dia é utilizado para testá-los. Métricas quantitativas de detecção de anomalias são calculadas e discutidas

para cada NIDS. Estão destacadas na tabela 3 as configurações de hardware e software do sistema computacional utilizado na realização dos experimentos.

Tabela 3 – Configuração de desenvolvimento.

Sistema Operacional	Linux Mint 20 Cinnamon
RAM	8 GB
Processador	Intel© Core™ i7-4510U CPU @ 2.00GHz × 2
Python	3.10.9
Tensorflow	2.11.0
Keras	2.11.0

5.4.1 Métricas de Detecção de Anomalias

Quando o sistema erroneamente infere a presença ou ausência de anomalias em uma observação estatística de tráfego, têm-se os chamados falsos positivos (FP) e falsos negativos (FN), respectivamente. Já quando o NIDS corretamente classifica uma instância de tráfego como anômala ou benigna, diz-se que foi obtido, respectivamente, um verdadeiro positivo (VP) ou verdadeiro negativo (VN). Esses quatro contadores que descrevem os acertos e erros do sistema são agrupados formando a chamada matriz de confusão (*confusion matrix*), como ilustrado na Figura 22. Essa é a métrica básica utilizada pela comunidade científica para medir e comparar o desempenho em detecção de anomalias de diferentes implementações NIDS para um mesmo conjunto de dados [140]. Outras métricas podem ser derivadas a partir da matriz de confusão, como por exemplo *Precision, Recall, F1-score, Matthews Correlation Coefficient (MCC), Receiver Operating Characteristic (ROC) curve* e *Area under the ROC Curve (AUROC)* [87], [88].

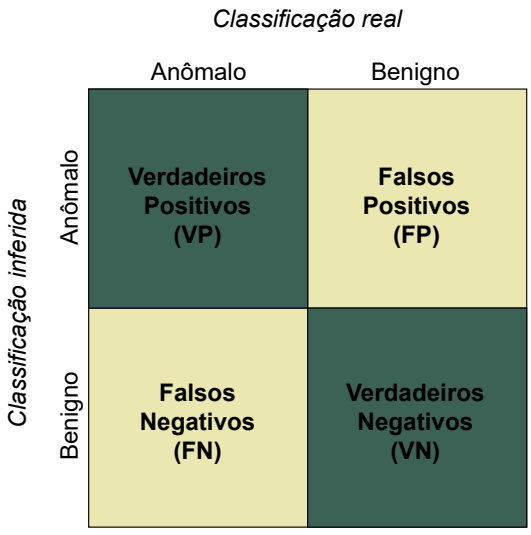

Figura 22 – Matriz de confusão.

A métrica *Precision* é definida pela equação 5.2. Ela representa a fração das observações indicadas como anômalas pelo sistema e que de fato são anormais. Quanto menos falsos positivos ocorrerem, melhor será o *Precision* do NIDS.

$$Precision = \frac{VP}{VP + FP} \tag{5.2}$$

O *Recall* pode ser calculado conforme a equação 5.3. Esse cálculo indica a fração de todas as observações anômalas que foram corretamente classificadas pelo sistema. Quanto menos falsos negativos o sistema apresentar, melhor será a sua pontuação em *Recall*.

$$Recall = \frac{VP}{VP + FN} \tag{5.3}$$

A média harmônica entre *Precision* e *Recall* é calculada pela métrica *F1-score* com base na equação 5.4. Ela produz valores no intervalo $[0,\ 1]$. Quanto maior o valor do *Precision* e *Recall* maior é o valor do F1-score, indicando que o sistema possui poucos falsos positivos e falsos negativos.

$$F1 - score = 2 * \frac{Precision\ *\ Recall}{Precision\ +\ Recall} \tag{5.4}$$

Uma métrica representativa e confiável do desempenho do modelo de classificação é a *Matthews Correlation Coefficient* (MCC), descrita pela equação 5.5. Essa métrica considera todas as quatro células da matriz de confusão no seu cálculo. Os valores possíveis para o MCC pertencem ao intervalo $[-1,\ 1]$. O valor 1 representa um classificador perfeito que identifica as observações corretamente. O resultado -1 corresponde a um modelo ótimo na tarefa de gerar saídas invertidas, classificando classes positivas como negativas e vice-versa. Um valor igual a 0 para o MCC indica que o sistema não aprendeu os padrões anômalos dos dados e classifica as observações aleatoriamente. Quanto menos falsos positivos e falsos negativos o sistema apresentar, melhor será a sua pontuação em MCC.

$$MCC = \frac{VP\ *\ VN -\ FP\ *\ FN}{\sqrt{(VP\ +\ FP)(VP\ +\ FN)(VN\ +\ FP)(VN\ +\ FN)}} \tag{5.5}$$

A curva ROC é comumente utilizada para avaliar modelos de classificação binária cuja saída representa a probabilidade de um exemplo ser considerado positivo. Esses modelos geralmente trabalham com um limiar de decisão que determina a probabilidade mínima para um exemplo ser considerado positivo. Nesse contexto, a curva ROC apresenta a comparação do *True Positive Rate (Recall)* e do *False Positive Rate (FPR)* apresentado pelo modelo para diferentes limiares [141], como ilustrado na Figura 23. A métrica False

Positive Rate indica a fração dos exemplos de dados negativos que foram erroneamente classificados como positivos. Quanto mais a curva estiver próxima do canto superior esquerdo do gráfico, melhor o desempenho geral do modelo na tarefa de classificação. Pois, ele pode obter um alto valor de *Recall* enquanto mantém o valor de FPR baixo.

O cálculo da área sob a curva ROC (AUROC) apresenta um resumo do desempenho do classificador [142]. Mais especificamente, ele representa a probabilidade de que o classificador produzirá saídas mais próximas de 1 para um exemplo positivo do que para um negativo. Quanto maior o valor da área, maior a distância que o modelo atribui entre dados anômalos e normais, facilitando a definição de um limiar de separação. O AUROC pode assumir valores no intervalo [0, 1]. Um valor igual ou inferior a 0,5 indica um classificador ruim.

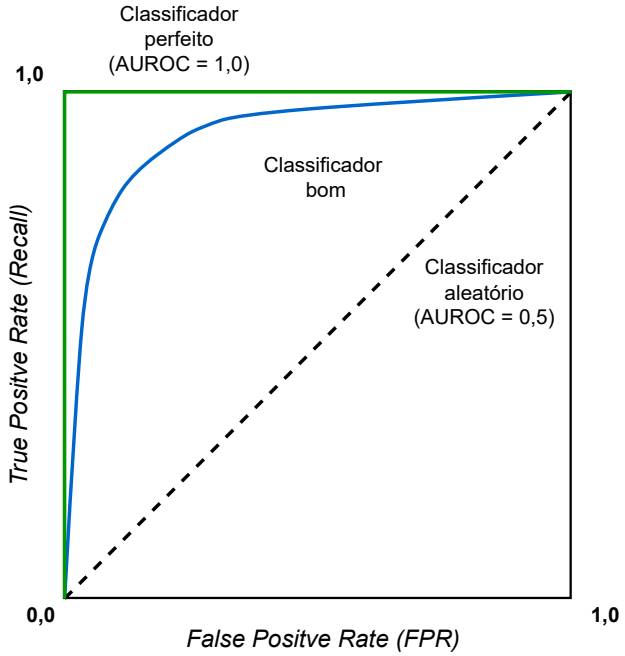

Figura 23 – Curva ROC.

5.4.2 NIDS-GAN

São ilustradas nas figuras 24 e 25 as arquiteturas das redes discriminadora e geradora que compõem a rede GAN. Essas redes foram implementadas utilizando o modelo de Rede Neural convencional. O discriminador possui duas camadas ocultas com 64 neurônios cada. O gerador também detêm duas camadas ocultas, as quais são formadas por 16 neurônios cada. Outros hiperparâmetros não relacionados à estrutura das redes são apresentados na tabela 4. Muitas das heurísticas descritas por Radford *et al.* [139] foram utilizadas nas definições de hiperparâmetros dessa rede, como: função de ativação *Leaky-ReLu* com inclinação 0,2 para as camadas ocultas de ambas as redes; função de ativação *tanh* para a camada de saída do gerador; função de ativação *sigmoid* para a camada

de saída do discriminador; espaço latente com distribuição Gaussiana; otimizador *Adam*. Outros hiperparâmetros foram calibrados por meio de múltiplos experimentos, como: camadas com *dropout*; número de camadas internas e neurônios de cada rede; número de dimensões do espaço latente; épocas de treinamento; tamanho do *mini-batch*; taxa de aprendizagem de cada rede; função de perda.

Tabela 4 – Hiperparâmetros não estruturais da rede GAN.

Hiperparâmetro	**Valor**
Número de épocas	20
Tamanho de *mini-batch*	256
Taxa de aprendizagem (discriminador)	0,00002
Taxa de aprendizagem (gerador)	0,0002
Função de perda	*binary cross entropy*
Otimizador	Adam

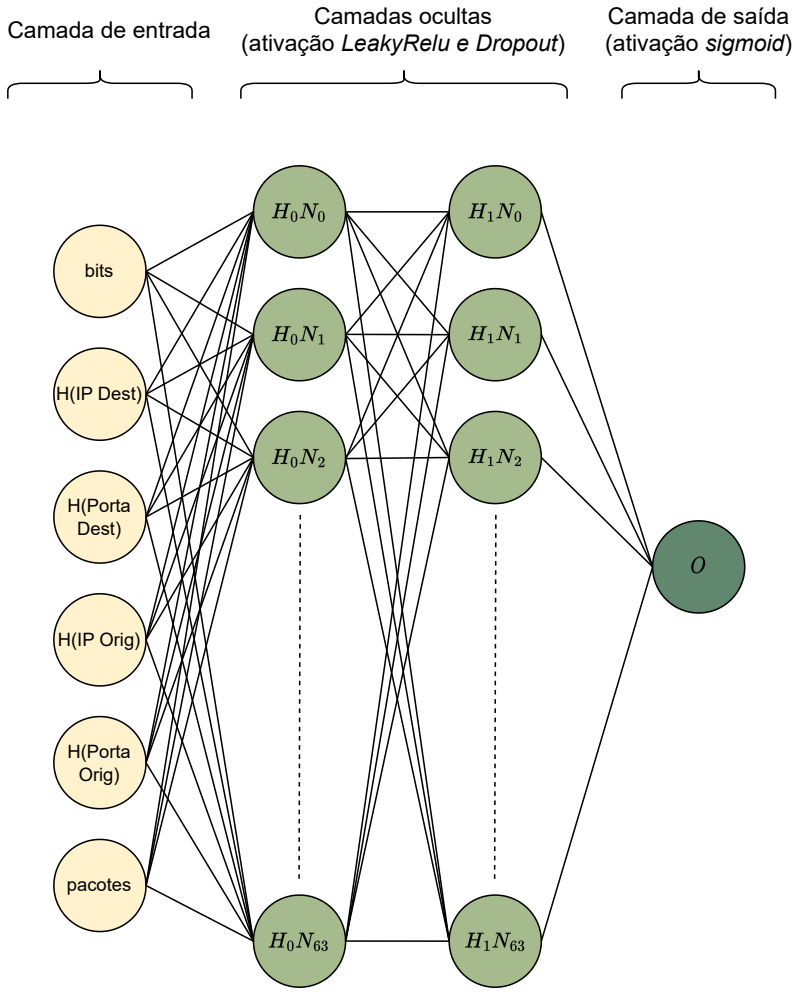

Figura 24 – Arquitetura da rede discriminadora.

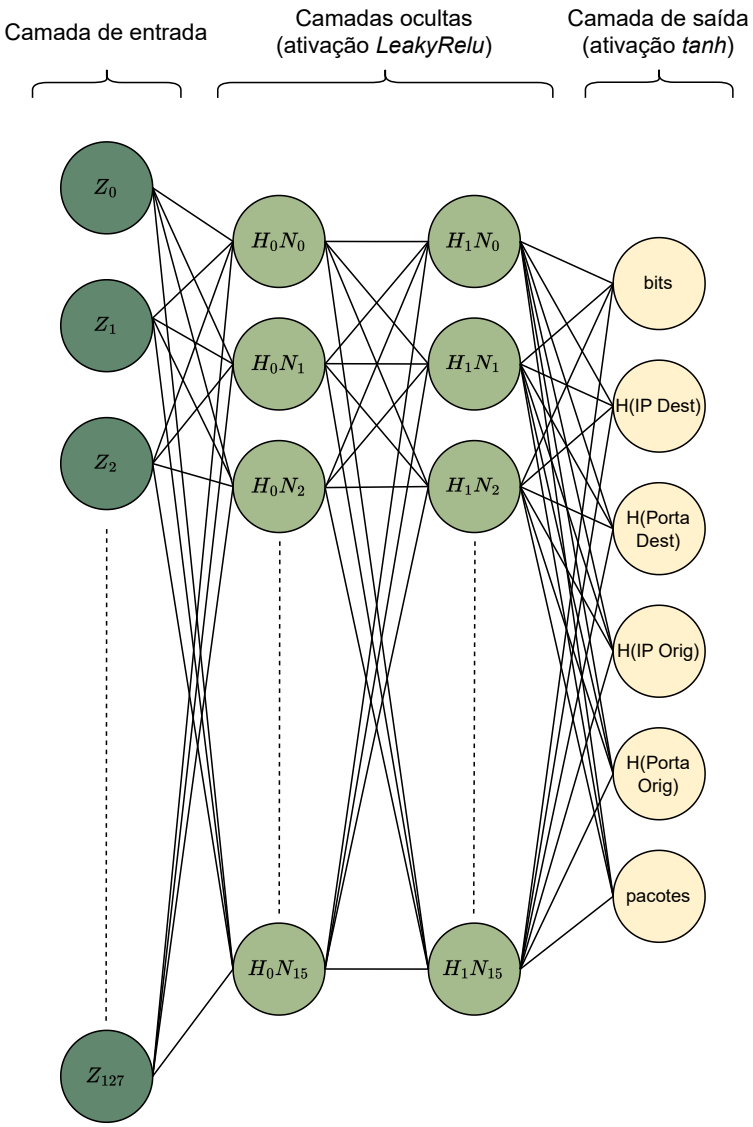

Figura 25 – Arquitetura da rede geradora.

O treinamento da rede GAN termina quando as saídas das funções de perda de cada rede se equilibram e permanecem próximas entre si. A Figura 26 apresenta o gráfico da perda de cada rede ao longo das iterações de treinamento. Pode-se notar que as funções estabilizam-se próximas ao valor 0,68 por volta da décima sétima iteração. Após o treinamento a rede GAN possui uma representação interna do *baseline* comportamental da rede. O gerador aprende a gerar uma distribuição próxima da distribuição original do tráfego, como ilustrado na Figura 27. O discriminador gera uma saída próxima de 0,4952 para observações benignas de tráfego, como indicado na Figura 28.

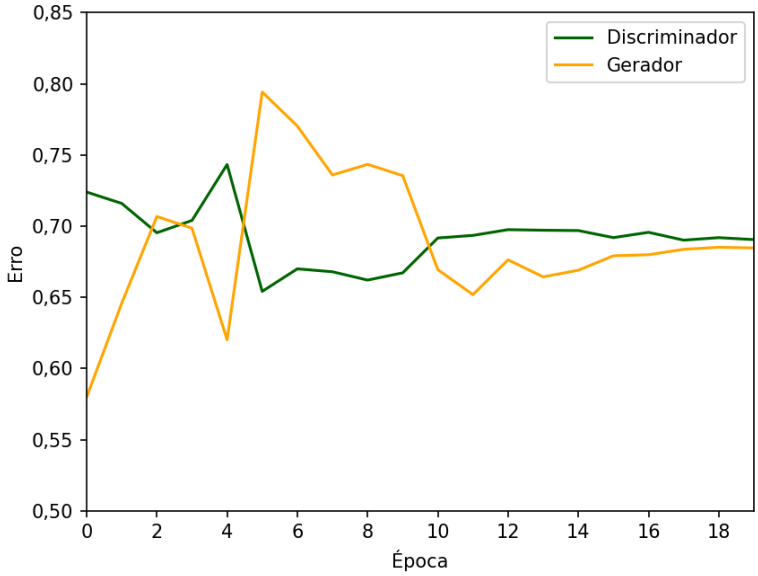

Figura 26 – Função de perda da rede GAN ao longo do treinamento.

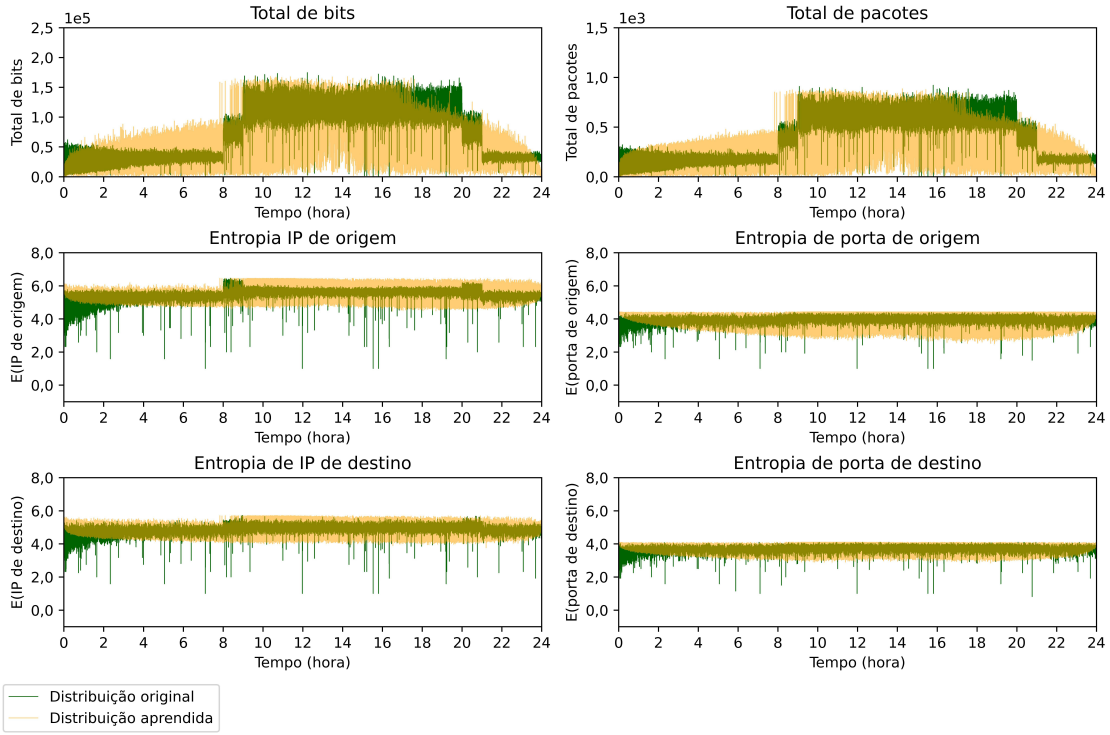

Figura 27 – Distribuição aprendida pelo gerador.

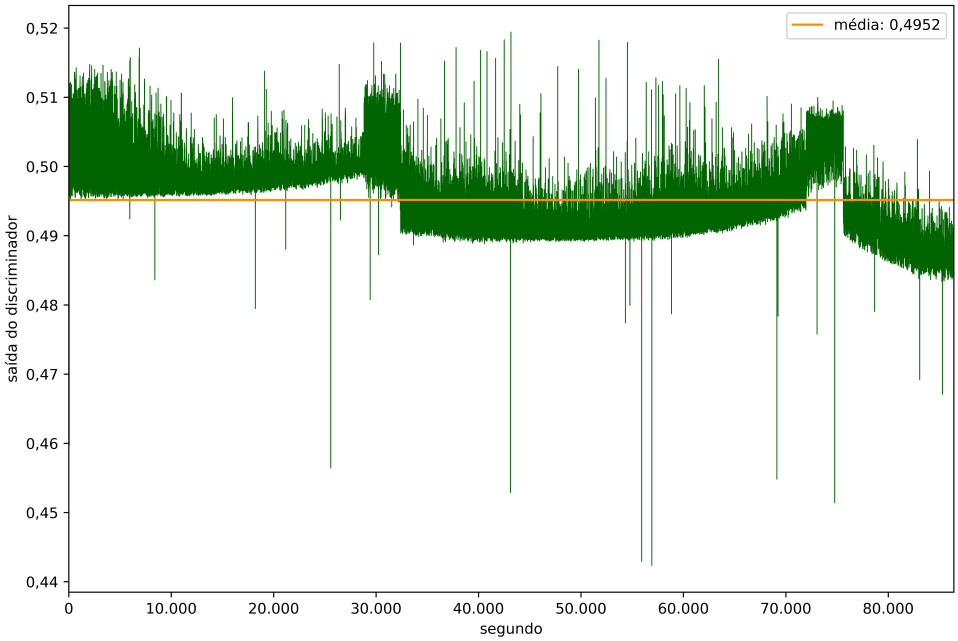

Figura 28 – Saída do discriminador para um conjunto de observações benignas (Dia 1 ORION).

O melhor limiar de decisão encontrado foi de 0,0232, provendo ao sistema uma pontuação de 0,9978 na métrica MCC. Esse é o valor ótimo de separação entre observações benignas das malignas para o dia 2 da base ORION. Como será apresentado a seguir, esse limiar mostra-se genérico e continua sendo representativo para um novo dia de tráfego analisado (dia 3 da base ORION).

5.4.3 NIDS-GRU

É ilustrada na Figura 29 a arquitetura da rede GRU. Ela possui duas camadas ocultas com 32 e 24 neurônios cada. Os demais hiperparâmetros que não se relacionam a estrutura da rede estão apresentados na tabela 5. Todos os hiperparâmetros foram ajustados mediante múltiplos experimentos.

Tabela 5 – Hiperparâmetros não estruturais da rede GRU.

Hiperparâmetro	**Valor**
Número de épocas	20
Tamanho de *mini-batch*	64
Taxa de aprendizagem	0,001
Função de perda	*mean squared error*
Otimizador	Adam
Tamanho de janela de dados	10

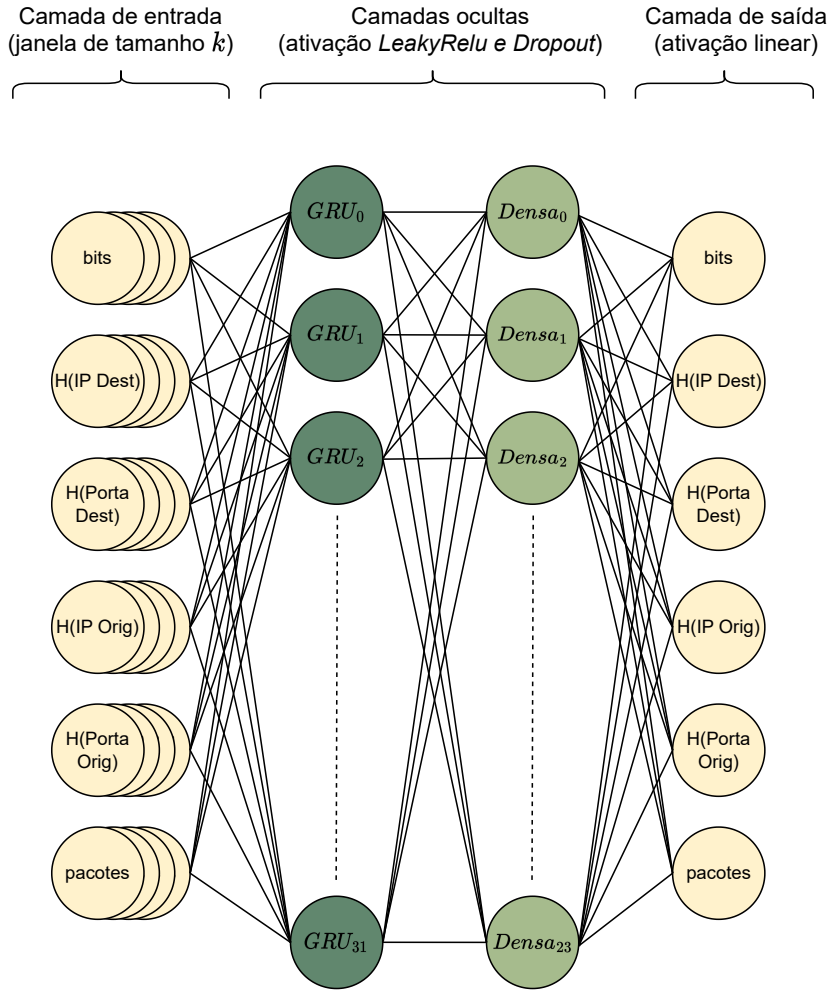

Figura 29 – Arquitetura da rede GRU.

A Figura 30 ilustra o gráfico das saídas da função de perda ao longo das iterações de treinamento da rede GRU. Pode-se notar que o ponto de convergência da rede é atingido com uma perda de aproximadamente 0,28 por volta da vigésima iteração. Após o treinamento, a rede GRU detém uma representação interna do *baseline* comportamental da rede. Essa rede aprende a prever o comportamento (valores numéricos) de observações de tráfego benignas com um erro mínimo, como ilustrado na Figura 31. Traçado em verde têm-se os comportamentos reais observados no tráfego de rede. A linha laranja representa os comportamentos previstos.

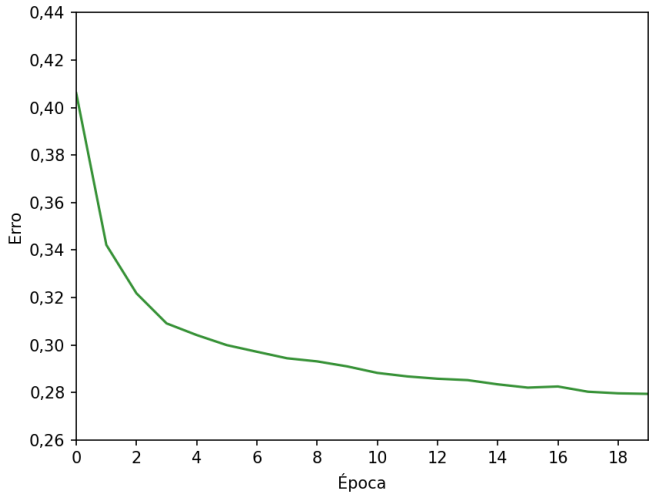

Figura 30 – Função de perda da rede GRU ao longo do treinamento.

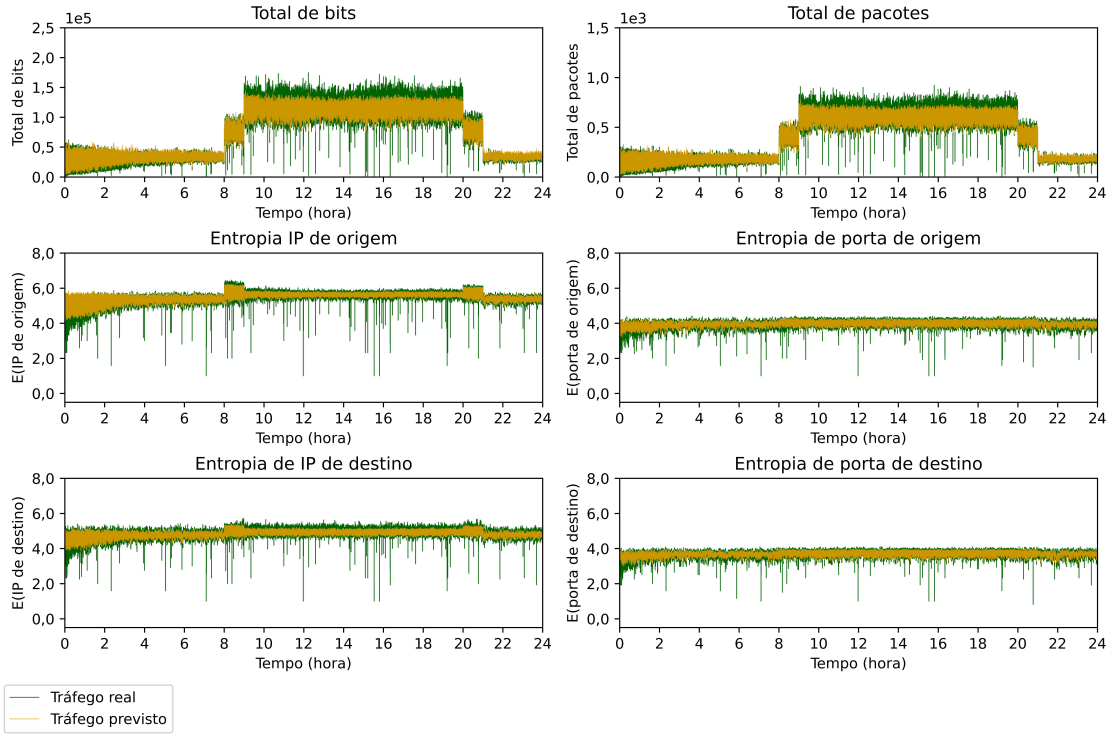

Figura 31 – Previsões de tráfego da rede GRU (Dia 1 ORION).

O limiar encontrado que melhor separa as observações anômalas das normais para o dia 2 da base ORION foi de 29,3969. Ele provê ao sistema um valor de 0,9946 na métrica MCC. Como será discutido a seguir, esse limiar continua útil na análise de outros conjuntos de dados (dia 3 da base ORION).

5.4.4 Comparação da Capacidade de Detecção de Anomalias dos Sistemas

A Figura 32 ilustra as saídas do discriminador para cada uma das observações de tráfego do dia 3 da base ORION. A reta sólida laranja representa a média da saída do discriminador para observações benignas (*baseline*). A reta tracejada indica o limiar de decisão. As observações para as quais o gerador gera uma saída acima do limiar são consideradas anômalas (rótulo 1). Observações cujas saídas estão abaixo da média são consideradas benignas. Pois, esse valor se aproxima de 0, o qual é o rótulo de normalidade utilizado no treinamento da rede GAN. A região destacada em vermelho indica o intervalo de observações malignas, as quais carregam traços de ataque DDoS. É possível notar que, em geral, a rede discriminadora produz saídas que se distanciam do *baseline* para esse intervalo.

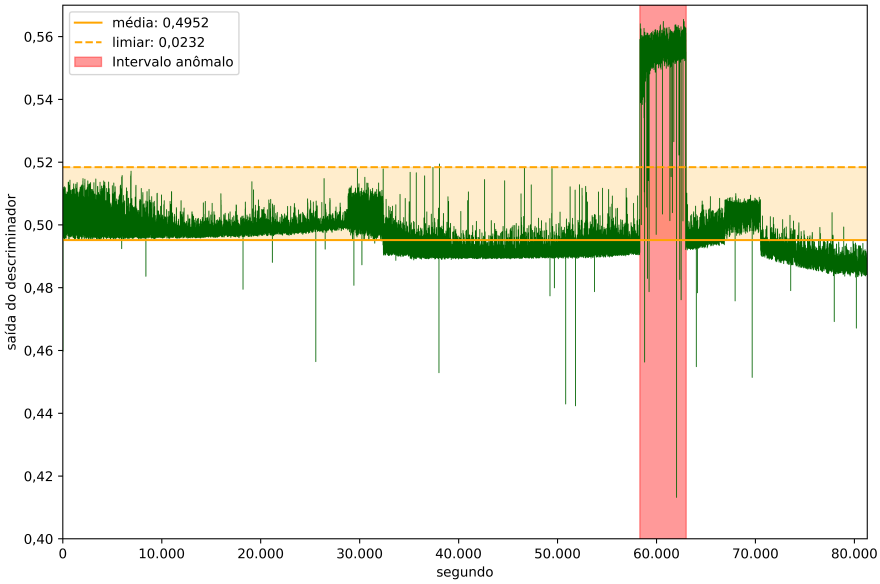

Figura 32 – Saída do discriminador para observações benignas e malignas (Dia 3 ORION).

A Figura 33 apresenta a matriz de confusão calculada a partir das inferências de rótulo realizadas pelo NIDS-GAN para o terceiro dia de tráfego da base ORION. Percebe-se que o sistema possui pouco erro. Ele obteve 19 falsos negativos, ou seja, classificou 19 observações maliciosas como normais. Além disso, ocorreu apenas 1 falso positivo, isto é, uma observação benigna classificada como anormal.

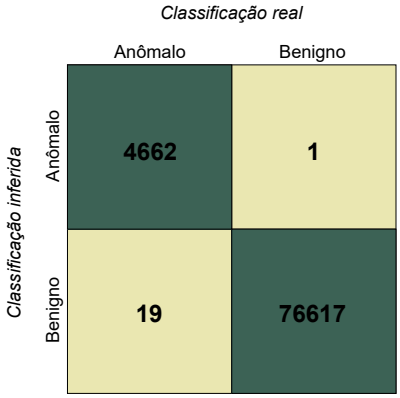

Figura 33 – Matriz de confusão para o dia 3 ORION (NIDS-GAN).

São apresentadas na Figura 34 as previsões comportamentais da rede GRU para cada um dos segundos do dia 3 da base ORION. A linha verde indica o tráfego real e a laranja o tráfego previsto. A rede GRU pode prever com alta precisão o comportamento de observações benignas. Nota-se que as previsões da rede possuem uma distância maior do tráfego real para o intervalo em vermelho, o qual contêm comportamento anômalo. É justamente nessas situações em que a diferença entre o comportamento previsto e o observado na rede ultrapassam o limiar de tolerância, fazendo com que a observação seja considerada anômala (rótulo 1).

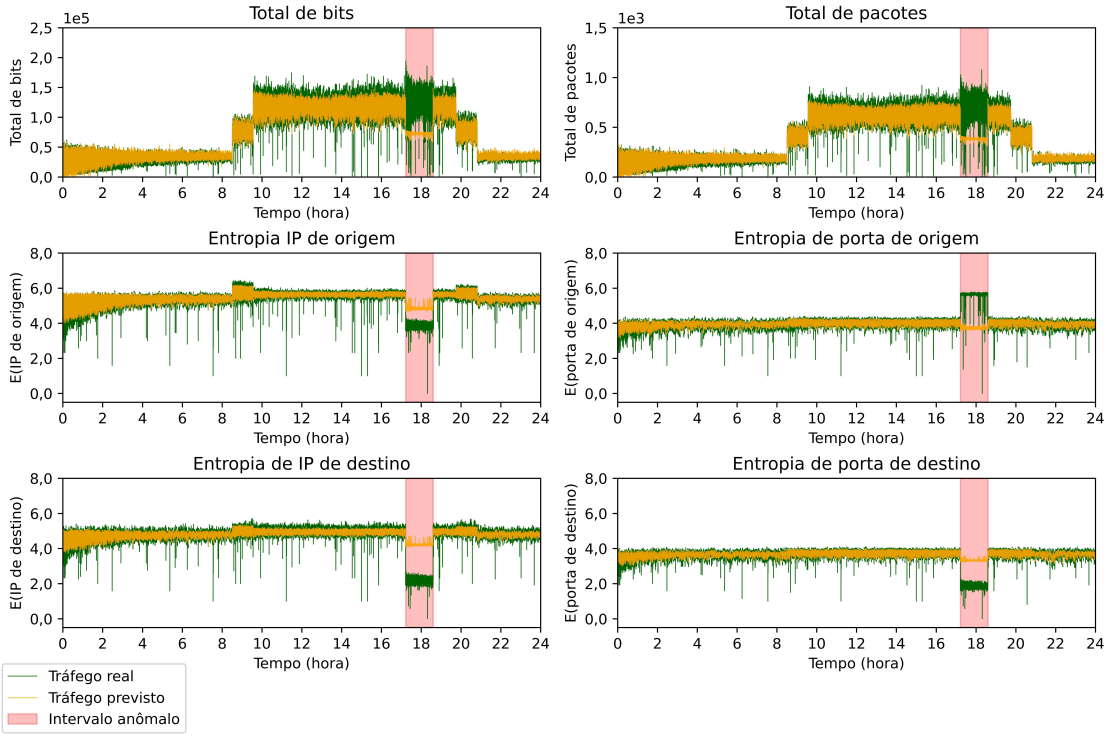

Figura 34 – Previsões de tráfego da rede GRU (Dia 3 ORION).

As inferências de rótulo geradas pelo NIDS-GRU foram agrupadas formando a matriz de confusão apresentada na Figura 35. Primeiramente, nota-se que o sistema apresenta 41 vezes mais falsos positivos que o NIDS-GAN. No entanto, esse sistema consegue obter menos falsos negativos que o seu concorrente.

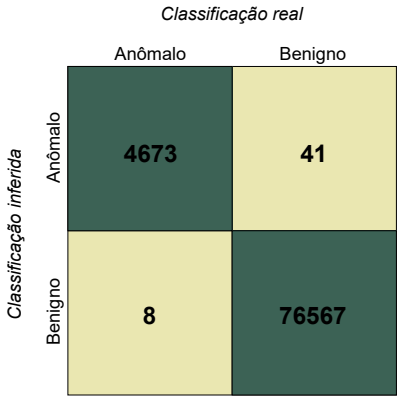

Figura 35 – Matriz de confusão para o dia 3 ORION (NIDS-GRU).

São ilustrados na Figura 36 os resultados do cálculo de métricas descritas na Seção 5.4.1. Essas métricas sumarizam o desempenho de cada NIDS na tarefa de detecção de anomalias. É notável que ambos os sistemas atingem acima de 99% de pontuação em cada uma das quatro métricas destacadas. A diferença na capacidade de detecção de anomalias dos sistemas pode ser observada apenas a partir da terceira casa decimal. O NIDS-GAN atinge um valor de 0,9997 na métrica *Precision* superando o seu concorrente que obteve 0,9913. Pode-se dizer que o primeiro sistema tende a acertar mais que o segundo ao classificar observações como anômalas. O NIDS-GRU possui o melhor valor na métrica *Recall* alcançando a marca de aproximadamente 0,9982 contra 0,9959 do NIDS-GAN. Esse resultado mostra que, em geral, o NIDS-GRU consegue identificar mais instâncias verdadeiramente anômalas que o outro sistema. Os valores das métricas *F1-score* e MCC indicam que o NIDS-GAN possui um melhor balanço entre falsos positivos e falsos negativos. O sistema baseado em rede GAN supera o seu concorrente obtendo, respectivamente, os valores 0,9978 e 0,9977 para essas métricas.

As curvas ROC para o NIDS-GAN e o NIDS-GRU são ilustradas na Figura 37. O bom desempenho apresentado pelas métricas mencionadas anteriormente se reflete na forma das curvas. Elas se aproximam do canto superior esquerdo, indicando que ambos sistemas possuem uma boa capacidade de discriminação entre observações normais e anômalas. Isto é, eles conseguem separar essas observações, possibilitando encontrar limiares precisos. A diferença entre os dois NIDS fica mais clara analisando o quadrado no centro da Figura 37, que apresenta uma visualização aproximada das curvas ROC. A curva para o NIDS-GRU se posiciona acima da curva do seu concorrente. De fato, o NIDS-GRU

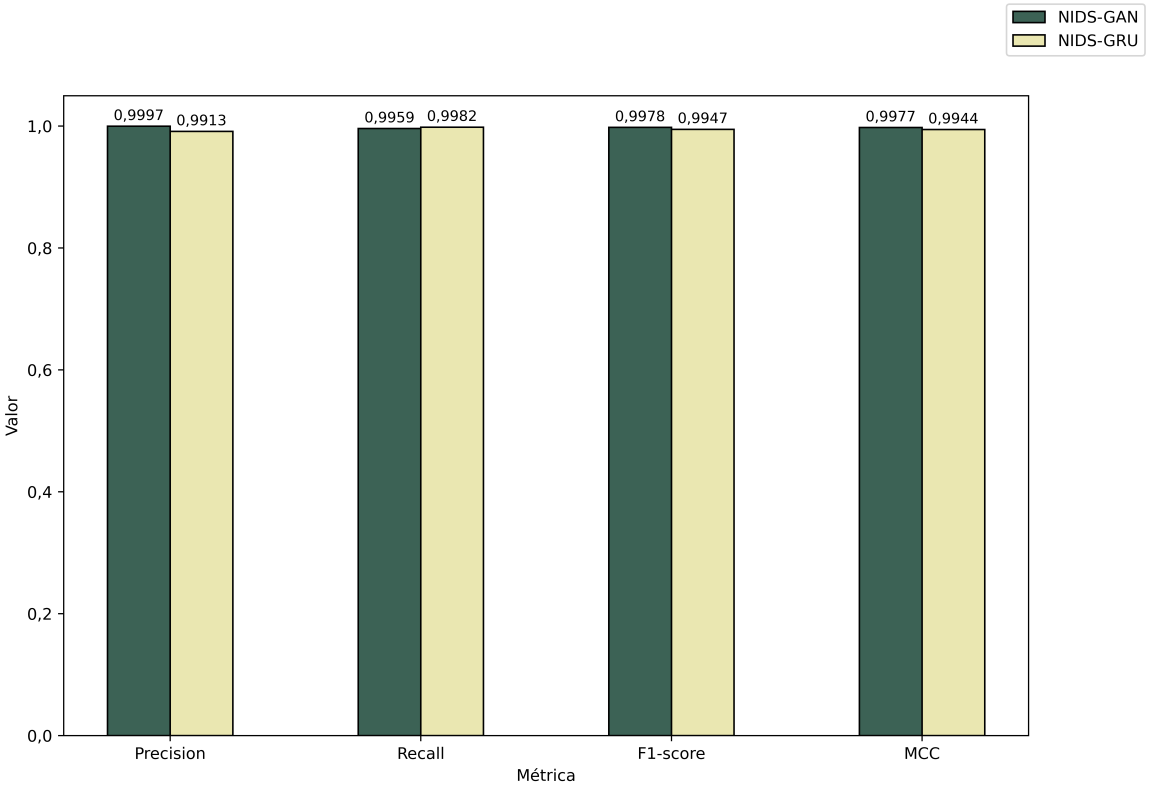

Figura 36 – Métricas de desempenho.

possui um AUROC de 0,9996 enquanto o NIDS-GAN possui 0,9984. Assim, pode-se concluir que o sistema baseado em GRU consegue gerar probabilidades mais distantes entre observações anômalas e normais do que o NIDS-GAN. Isso facilita a definição de limiares de decisão para separar e classificar esses dados.

Apesar de o sistema GAN atribuir distâncias menores entre dados anômalos e normais, ele ainda consegue definir um limiar de separação preciso para esses dados. Essa conclusão é suportada pelos valores superiores que o sistema obteve para as métricas *F1-score* e MCC, obtendo um melhor balanço entre falsos positivos e falsos negativos. Portanto, pode-se dizer que o NIDS-GAN apresenta um desempenho geral superior ao do NIDS-GRU para o cenário de execução considerado.

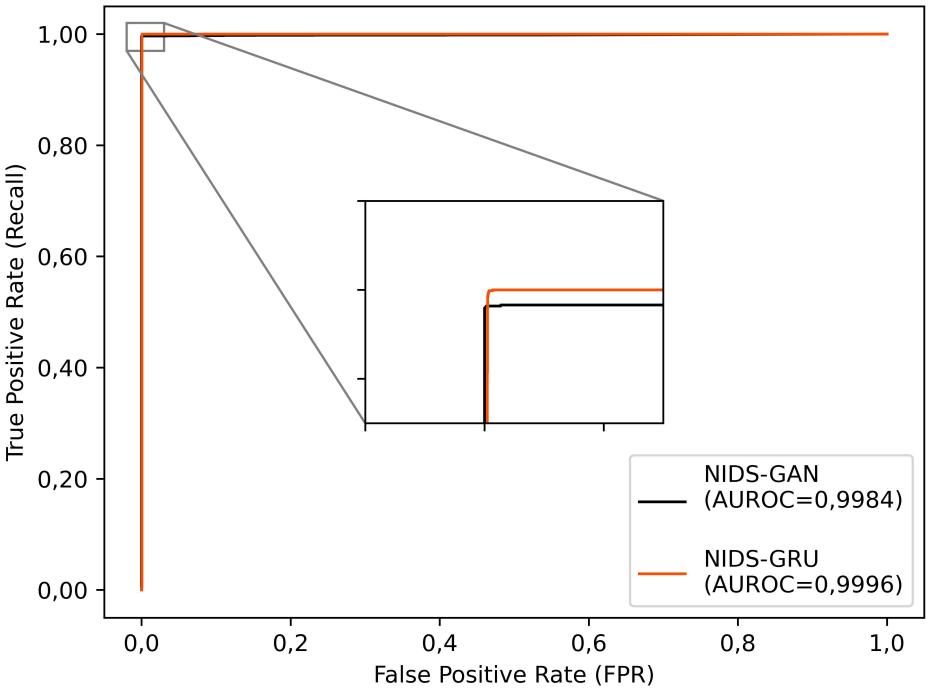

Figura 37 – Curvas ROC para os sistemas.

6 CONCLUSÃO

Foi apresentado um estudo da aplicabilidade de Rede Adversária Generativa na implementação de um Sistema de Detecção de Intrusão para redes SDN. O sistema desenvolvido é composto de quatro módulos: coleta de tráfego, pré-processamento de dados, caracterização de tráfego e detecção de anomalias. A rede GAN foi utilizada para implementar o módulo caracterizador de tráfego que aprende o perfil de comportamento normal da rede (*baseline*). O NIDS realiza a coleta e análise de dados de tráfego segundo a segundo, gerando um alerta quando a presença de ataques é identificada.

A capacidade de detecção de anomalias do sistema implementado foi testada utilizando uma base de dados coletada de uma rede SDN emulada. Esse conjunto de dados é formado por um dia de tráfego sem ataques e dois dias com ataques DDoS. Os dois primeiros dias foram utilizados para configurar o sistema e o terceiro dia para a realização de testes. Para medir o desempenho do NIDS foram utilizadas métricas quantitativas, como *Precision, Recall, F1-score, Matthews Correlation Coefficient* (MCC), *ROC curve* e *Area under the ROC curve*.

O desempenho do NIDS baseado em rede GAN foi comparado com o de um sistema semelhante construído com base em Rede Neural Recorrente GRU. A rede GRU é conhecida pela sua adequabilidade no processamento de séries temporais, como dados de tráfego de rede. Os resultados mostram que ambos os sistemas conseguem identificar aproximadamente 99% dos ataques DDoS lançados à rede emulada. As duas implementações podem discriminar precisamente entre tráfego normal e tráfego contaminado com ataques DDoS. O NIDS implementado com rede GAN obtém as pontuações 0,9978 e 0,9977 nas métricas *F1-score* e MCC, respectivamente. Já o sistema desenvolvido com rede GRU atinge as pontuações de 0,9947 e 0,9944. Esses resultados sugerem que o sistema baseado em rede GAN obtém um melhor balanço entre falsos positivos e falsos negativos do que o seu concorrente. Assim, o NIDS-GAN supera o NIDS-GRU no cenário de execução considerado. A partir dos resultados, conclui-se que o modelo de Rede Adversária Generativa é uma alternativa viável na implementação de um sistema de detecção de intrusão de redes.

Como trabalho futuro destaca-se a utilização de métricas adicionais para auxiliar na comparação dos sistemas desenvolvidos, como complexidade computacional e escalabilidade. Também deseja-se testar o sistema NIDS-GAN utilizando outras bases de dados com o intuito de verificar o seu desempenho em diferentes cenários de execução. Pretende-se expandir as capacidades do NIDS-GAN implementado. O novo sistema será capaz não só de detectar a ocorrência de ataques, mas também identificar o tipo de ataque e mitigá-lo de maneira automática de modo a manter a rede funcionado corretamente. Planeja-se implementar as redes discriminadora e geradora utilizando modelos de Aprendizado Pro-

fundo, como CNN, LSTM e GRU para verificar o impacto no desempenho do NIDS.

As contribuições deste trabalho são:

- Apresentar uma revisão da literatura sobre detecção de anomalias baseada em Aprendizado Profundo.
- Demonstrar o uso do modelo de Rede Adversária Generativa em uma aplicação do mundo real.
- Comparar o desempenho em detecção de anomalias do modelo de Rede Adversária Generativa com o modelo de Rede Neural Profunda Recorrente GRU utilizando métricas quantitativas a partir de uma base de dados comum.

REFERÊNCIAS

[1] BRASIL, B. C. do. *O que é Pix?* <https://www.bcb.gov.br/estabilidadefinanceira/pix>. Accessed: 2022-08-19.

[2] YANG, L.; MOUBAYED, A.; SHAMI, A. Mth-ids: a multitiered hybrid intrusion detection system for internet of vehicles. *IEEE Internet of Things Journal*, IEEE, v. 9, n. 1, p. 616–632, 2021. Disponível em: <https://doi.org/10.1109/JIOT.2021.3084796>.

[3] TAMA, B. A.; COMUZZI, M.; RHEE, K.-H. Tse-ids: A two-stage classifier ensemble for intelligent anomaly-based intrusion detection system. *IEEE access*, IEEE, v. 7, p. 94497–94507, 2019. Disponível em: <https://doi.org/10.1109/ACCESS.2019.2928048>.

[4] ESKANDARI, M. et al. Passban ids: An intelligent anomaly-based intrusion detection system for iot edge devices. *IEEE Internet of Things Journal*, IEEE, v. 7, n. 8, p. 6882–6897, 2020. Disponível em: <https://doi.org/10.1109/JIOT.2020.2970501>.

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

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

[7] LENT, D. M. B. et al. A gated recurrent unit deep learning model to detect and mitigate distributed denial of service and portscan attacks. *IEEE Access*, IEEE, v. 10, p. 73229–73242, 2022. Disponível em: <https://doi.org/10.1109/ACCESS.2022.3190008>.

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

[9] ASSIS, M. V. 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*, IEEE, v. 5, p. 9485–9496, 2017. Disponível em: <https://doi.org/10.1109/ACCESS.2017.2702341>.

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

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

[12] ANTHI, E. et al. A supervised intrusion detection system for smart home iot devices. *IEEE Internet of Things Journal*, IEEE, v. 6, n. 5, p. 9042–9053, 2019. Disponível em: <https://doi.org/10.1109/JIOT.2019.2926365>.

[13] PÉREZ, S. I.; MORAL-RUBIO, S.; CRIADO, R. A new approach to combine multiplex networks and time series attributes: Building intrusion detection systems (ids) in cybersecurity. *Chaos, Solitons & Fractals*, Elsevier, v. 150, p. 111143, 2021. Disponível em: <https://doi.org/10.1016/j.chaos.2021.111143>.

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

[15] 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>.

[16] GUPTA, N.; JINDAL, V.; BEDI, P. Lio-ids: handling class imbalance using lstm and improved one-vs-one technique in intrusion detection system. *Computer Networks*, Elsevier, v. 192, p. 108076, 2021. Disponível em: <https://doi.org/10.1016/j.comnet.2021.108076>.

[17] BEDI, P.; GUPTA, N.; JINDAL, V. Siam-ids: Handling class imbalance problem in intrusion detection systems using siamese neural network. *Procedia Computer Science*, Elsevier, v. 171, p. 780–789, 2020. Disponível em: <https://doi.org/10.1016/j.procs.2020.04.085>.

[18] BALKANINSIGHT. *Albania Blames 'Massive Cyber Attack' as Govt Servers go Down*. <https://balkaninsight.com/2022/07/18/albania-gov-says-it-is-being-attacked-as-service-servers-are-down/>. Accessed: 2022-08-19.

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

[20] GAO, X. et al. An adaptive ensemble machine learning model for intrusion detection. *IEEE Access*, IEEE, v. 7, p. 82512–82521, 2019. Disponível em: <https://doi.org/10.1109/ACCESS.2019.2923640>.

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

[22] JAN, S. U. et al. Toward a lightweight intrusion detection system for the internet of things. *IEEE Access*, IEEE, v. 7, p. 42450–42471, 2019. Disponível em: <https://doi.org/10.1109/ACCESS.2019.2907965>.

[23] LO, W. et al. A hybrid deep learning based intrusion detection system using spatial-temporal representation of in-vehicle network traffic. *Vehicular Communications*, Elsevier, v. 35, p. 100471, 2022. Disponível em: <https://doi.org/10.1016/j.vehcom.2022.100471>.

[24] RAO, K. N.; RAO, K. V.; PVGD, P. R. A hybrid intrusion detection system based on sparse autoencoder and deep neural network. *Computer Communications*, Elsevier, v. 180, p. 77–88, 2021. Disponível em: <https://doi.org/10.1016/j.comcom.2021.08.026>.

[25] DINA, A. S.; MANIVANNAN, D. Intrusion detection based on machine learning techniques in computer networks. *Internet of Things*, Elsevier, v. 16, p. 100462, 2021. Disponível em: <https://doi.org/10.1016/j.iot.2021.100462>.

[26] QIU, W. et al. Hybrid intrusion detection system based on dempster-shafer evidence theory. *Computers & Security*, Elsevier, v. 117, p. 102709, 2022. Disponível em: <https://doi.org/10.1016/j.cose.2022.102709>.

[27] ELSAYED, M. S. et al. A novel hybrid model for intrusion detection systems in sdns based on cnn and a new regularization technique. *Journal of Network and Computer Applications*, Elsevier, v. 191, p. 103160, 2021. Disponível em: <https://doi.org/10.1016/j.jnca.2021.103160>.

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

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

[30] ALDWEESH, A.; DERHAB, A.; EMAM, A. Z. Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues. *Knowledge-Based Systems*, Elsevier, v. 189, p. 105124, 2020. Disponível em: <https://doi.org/10.1016/j.knosys.2019.105124>.

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

[32] NAVIDAN, H. et al. Generative adversarial networks (gans) in networking: A comprehensive survey & evaluation. *Computer Networks*, Elsevier, v. 194, p. 108149, 2021. Disponível em: <https://doi.org/10.1016/j.comnet.2021.108149>.

[33] LEE, S.-W. et al. Towards secure intrusion detection systems using deep learning techniques: Comprehensive analysis and review. *Journal of Network and Computer Applications*, Elsevier, v. 187, p. 103111, 2021. Disponível em: <https://doi.org/10.1016/j.jnca.2021.103111>.

[34] LAZAREVIC, A. et al. A comparative study of anomaly detection schemes in network intrusion detection. In: SIAM. *Proceedings of the 2003 SIAM international conference on data mining*. 2003. p. 25–36. Disponível em: <https://doi.org/10.1137/1.9781611972733.3>.

[35] PATCHA, A.; PARK, J.-M. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer networks*, Elsevier, v. 51, n. 12, p. 3448–3470, 2007. Disponível em: <https://doi.org/10.1016/j.comnet.2007.02.001>.

[36] KWON, D. et al. A survey of deep learning-based network anomaly detection. *Cluster Computing*, Springer, v. 22, n. 1, p. 949–961, 2019. Disponível em: <https://doi.org/10.1007/s10586-017-1117-8>.

[37] YANG, Z. et al. A systematic literature review of methods and datasets for anomaly-based network intrusion detection. *Computers & Security*, Elsevier, p. 102675, 2022. Disponível em: <https://doi.org/10.1016/j.cose.2022.102675>.

[38] HAJIHEIDARI, S. et al. Intrusion detection systems in the internet of things: A comprehensive investigation. *Computer Networks*, Elsevier, v. 160, p. 165–191, 2019. Disponível em: <https://doi.org/10.1016/j.comnet.2019.05.014>.

[39] LATA, S.; SINGH, D. Intrusion detection system in cloud environment: Literature survey & future research directions. *International Journal of Information Management Data Insights*, Elsevier, v. 2, n. 2, p. 100134, 2022. Disponível em: <https://doi.org/10.1016/j.jjimei.2022.100134>.

[40] KHAN, A. R. et al. Deep learning for intrusion detection and security of internet of things (iot): current analysis, challenges, and possible solutions. *Security and Communication Networks*, Hindawi, v. 2022, 2022. Disponível em: <ttps://doi.org/10.1155/2022/4016073>.

[41] SAHAY, R.; MENG, W.; JENSEN, C. D. The application of software defined networking on securing computer networks: A survey. *Journal of Network and Computer Applications*, Elsevier, v. 131, p. 89–108, 2019. Disponível em: <https://doi.org/10.1016/j.jnca.2019.01.019>.

[42] LI, W.; MENG, W.; KWOK, L. F. A survey on openflow-based software defined networks: Security challenges and countermeasures. *Journal of Network and Computer Applications*, Elsevier, v. 68, p. 126–139, 2016. Disponível em: <https://doi.org/10.1016/j.jnca.2016.04.011>.

[43] NISAR, K. et al. A survey on the architecture, application, and security of software defined networking: Challenges and open issues. *Internet of Things*, Elsevier, v. 12, p. 100289, 2020. Disponível em: <https://doi.org/10.1016/j.iot.2020.100289>.

[44] NUNES, B. A. A. et al. A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications surveys & tutorials*, IEEE, v. 16, n. 3, p. 1617–1634, 2014. Disponível em: <https://doi.org/10.1109/SURV.2014.012214.00180>.

[45] VALDOVINOS, I. A. et al. Emerging ddos attack detection and mitigation strategies in software-defined networks: Taxonomy, challenges and future directions. *Journal of Network and Computer Applications*, Elsevier, v. 187, p. 103093, 2021. Disponível em: <https://doi.org/10.1016/j.jnca.2021.103093>.

[46] MASOUDI, R.; GHAFFARI, A. Software defined networks: A survey. *Journal of Network and computer Applications*, Elsevier, v. 67, p. 1–25, 2016. Disponível em: <https://doi.org/10.1016/j.jnca.2016.03.016>.

[47] YUNGAICELA-NAULA, N. M. et al. Towards security automation in software defined networks. *Computer Communications*, Elsevier, v. 183, p. 64–82, 2022. Disponível em: <https://doi.org/10.1016/j.comcom.2021.11.014>.

[48] KREUTZ, D. et al. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, Ieee, v. 103, n. 1, p. 14–76, 2014. Disponível em: <https://doi.org/10.1109/JPROC.2014.2371999>.

[49] RING, M. et al. A survey of network-based intrusion detection data sets. *Computers & Security*, Elsevier, v. 86, p. 147–167, 2019. Disponível em: <https://doi.org/10.1016/j.cose.2019.06.005>.

[50] 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>.

[51] PANG, G. et al. Deep learning for anomaly detection: A review. *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, v. 54, n. 2, p. 1–38, 2021. Disponível em: <https://doi.org/10.1145/3439950>.

[52] SABUHI, M. et al. Applications of generative adversarial networks in anomaly detection: A systematic literature review. *IEEE Access*, IEEE, 2021. Disponível em: <https://doi.org/10.1109/ACCESS.2021.3131949>.

[53] XIN, Y. et al. Machine learning and deep learning methods for cybersecurity. *Ieee access*, IEEE, v. 6, p. 35365–35381, 2018. Disponível em: <https://doi.org/10.1109/ACCESS.2018.2836950>.

[54] PROENÇA, M. L.; ZARPELÃO, B. B.; MENDES, L. S. Anomaly detection for network servers using digital signature of network segment. In: IEEE. *Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop (AICT/SAPIR/ELETE'05)*. 2005. p. 290–295. Disponível em: <https://doi.org/10.1109/AICT.2005.26>.

[55] PROENÇA, M. L. et al. The hurst parameter for digital signature of network segment. In: SPRINGER. *International Conference on Telecommunications*. 2004. p. 772–781. Disponível em: <https://doi.org/10.1007/978-3-540-27824-5_103>.

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

[57] ASSIS, M. V. D.; RODRIGUES, J. J.; PROENÇA, M. L. A novel anomaly detection system based on seven-dimensional flow analysis. In: IEEE. *2013 IEEE Global Communications Conference (GLOBECOM)*. 2013. p. 735–740. Disponível em: <https://doi.org/10.1109/GLOCOM.2013.6831160>.

[58] JR, M. L. P. et al. Digital signature to help network management using flow analysis. *International Journal of Network Management*, Wiley Online Library, v. 26, n. 2, p. 76–94, 2016. Disponível em: <https://doi.org/10.1002/nem.1892>.

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

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

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

[62] BIJONE, M. A survey on secure network: intrusion detection & prevention approaches. *American Journal of Information Systems*, v. 4, n. 3, p. 69–88, 2016.

[63] MOHAMMED, M.; KHAN, M. B.; BASHIER, E. B. M. *Machine learning: algorithms and applications*. Crc Press, 2016. Disponível em: <https://doi.org/10.1201/9781315371658>.

[64] ALPAYDIN, E. *Introduction to machine learning*. [S.l.]: MIT press, 2020. ISBN 026201243X.

[65] GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep learning*. MIT press, 2016. Disponível em: <https://doi.org/10.4258/hir.2016.22.4.351>.

[66] LOURIDAS, P.; EBERT, C. Machine learning. *IEEE Software*, v. 33, p. 110–115, 09 2016. Disponível em: <https://doi.org/10.1109/MS.2016.114>.

[67] JORDAN, M. I.; MITCHELL, T. M. Machine learning: Trends, perspectives, and prospects. *Science*, American Association for the Advancement of Science, v. 349, n. 6245, p. 255–260, 2015. Disponível em: <https://doi.org/10.1126/science.aaa8415>.

[68] HATCHER, W. G.; YU, W. A survey of deep learning: Platforms, applications and emerging research trends. *IEEE Access*, IEEE, v. 6, p. 24411–24432, 2018. Disponível em: <https://doi.org/10.1109/ACCESS.2018.2830661>.

[69] XIE, J. et al. A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges. *IEEE Communications Surveys & Tutorials*, IEEE, v. 21, n. 1, p. 393–430, 2018. Disponível em: <https://doi.org/10.1109/COMST.2018.2866942>.

[70] SHRESTHA, A.; MAHMOOD, A. Review of deep learning algorithms and architectures. *IEEE access*, IEEE, v. 7, p. 53040–53065, 2019. Disponível em: <https://doi.org/10.1109/ACCESS.2019.2912200>.

[71] LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *nature*, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015. Disponível em: <https://doi.org/10.1038/nature14539>.

[72] YU, Y. et al. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , v. 31, n. 7, p. 1235–1270, 2019. Disponível em: <https://doi.org/10.1162/neco_a_01199>.

[73] DEY, R.; SALEM, F. M. Gate-variants of gated recurrent unit (gru) neural networks. In: IEEE. *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*. 2017. p. 1597–1600. Disponível em: <https://doi.org/10.1109/MWSCAS.2017.8053243>.

[74] HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, MIT Press, v. 9, n. 8, p. 1735–1780, 1997. Disponível em: <https://doi.org/10.1162/neco.1997.9.8.1735>.

[75] CHO, K. et al. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. Disponível em: <https://doi.org/10.48550/arXiv.1406.1078>.

[76] KHAN, A. et al. A survey of the recent architectures of deep convolutional neural networks. *Artificial intelligence review*, Springer, v. 53, n. 8, p. 5455–5516, 2020. Disponível em: <https://doi.org/10.1007/s10462-020-09825-6>.

[77] SAHA, S. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>. Accessed: 2023-01-21.

[78] ALBAWI, S.; MOHAMMED, T. A.; AL-ZAWI, S. Understanding of a convolutional neural network. In: IEEE. *2017 international conference on engineering and technology (ICET)*. 2017. p. 1–6. Disponível em: <https://doi.org/10.1109/ICEngTechnol.2017.8308186>.

[79] O'SHEA, K.; NASH, R. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015. Disponível em: <https://doi.org/10.48550/arXiv.1511.08458>.

[80] GU, J. et al. Recent advances in convolutional neural networks. *Pattern recognition*, Elsevier, v. 77, p. 354–377, 2018. Disponível em: <https://doi.org/10.1016/j.patcog.2017.10.013>.

[81] BAI, S.; KOLTER, J. Z.; KOLTUN, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018. Disponível em: <https://doi.org/10.48550/arXiv.1803.01271>.

[82] DUN, A.; YANG, Y.; LEI, F. Dynamic graph convolution neural network based on spatial-temporal correlation for air quality prediction. *Ecological Informatics*, Elsevier, v. 70, p. 101736, 2022. Disponível em: <https://doi.org/10.1016/j.ecoinf.2022.101736>.

[83] DUDUKCU, H. V.; TASKIRAN, M.; KAHRAMAN, N. Instantaneous power consumption prediction with modified temporal convolutional network for uavs. In: IEEE. *2022 45th International Conference on Telecommunications and Signal Processing (TSP)*. 2022. p. 106–109. Disponível em: <https://doi.org/10.1109/TSP55681.2022.9851230>.

[84] HEWAGE, P. et al. Temporal convolutional neural (tcn) network for an effective weather forecasting using time-series data from the local weather station. *Soft Computing*, Springer, v. 24, n. 21, p. 16453–16482, 2020. Disponível em: <https://doi.org/10.1007/s00500-020-04954-0>.

[85] ZHEN, Y. et al. Temporal convolution network based on attention mechanism for well production prediction. *Journal of Petroleum Science and Engineering*, Elsevier, v. 218, p. 111043, 2022. Disponível em: <https://doi.org/10.1016/j.petrol.2022.111043>.

[86] PENG, S.; ZHANG, J. A temporal convolutional network based method for fault diagnosis of deh system. In: IEEE. *2021 International Conference on Communications, Information System and Computer Engineering (CISCE)*. 2021. p. 863–867. Disponível em: <https://doi.org/10.1109/CISCE52179.2021.9445962>.

[87] CHOOBDAR, P.; NADERAN, M.; NADERAN, M. Detection and multi-class classification of intrusion in software defined networks using stacked auto-encoders and cicids2017 dataset. *Wireless Personal Communications*, Springer, p. 1–35, 2022. Disponível em: <https://doi.org/10.1007/s11277-021-09139-y>.

[88] PARK, S.; KIM, M.; LEE, S. Anomaly detection for http using convolutional autoencoders. *IEEE Access*, IEEE, v. 6, p. 70884–70901, 2018. Disponível em: <https://doi.org/10.1109/ACCESS.2018.2881003>.

[89] KAUSHAL, A. *Machine Learning: Cost Functions and Gradient Descent*. <https://ai.plainenglish.io/machine-learning-cost-functions-and-gradient-descent-6b6f9c355326>. Accessed: 2023-02-02.

[90] WEERTS, H. J.; MUELLER, A. C.; VANSCHOREN, J. Importance of tuning hyperparameters of machine learning algorithms. *arXiv preprint arXiv:2007.07588*, 2020. Disponível em: <https://doi.org/10.48550/arXiv.2007.07588>.

[91] PROBST, P.; BOULESTEIX, A.-L.; BISCHL, B. Tunability: Importance of hyperparameters of machine learning algorithms. *The Journal of Machine Learning Research*, JMLR. org, v. 20, n. 1, p. 1934–1965, 2019. Disponível em: <http://jmlr.org/papers/v20/18-444.html>.

[92] YANG, L.; SHAMI, A. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, Elsevier, v. 415, p. 295–316, 2020. Disponível em: <https://doi.org/10.1016/j.neucom.2020.07.061>.

[93] LIAO, L. et al. An empirical study of the impact of hyperparameter tuning and model optimization on the performance properties of deep neural networks. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, ACM New York, NY, v. 31, n. 3, p. 1–40, 2022. Disponível em: <https://doi.org/10.1145/3506695>.

[94] SMITH, L. N. A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay. *CoRR*, abs/1803.09820, 2018. Disponível em: <http://arxiv.org/abs/1803.09820>.

[95] FEURER, M.; HUTTER, F. Hyperparameter optimization. *Automated machine learning: Methods, systems, challenges*, Springer International Publishing, p. 3–33, 2019. Disponível em: <https://doi.org/10.1007/978-3-030-05318-5_1>.

[96] NAKISA, B. et al. Long short term memory hyperparameter optimization for a neural network based emotion recognition framework. *IEEE Access*, IEEE, v. 6, p. 49325–49338, 2018. Disponível em: <https://doi.org/10.1109/ACCESS.2018.2868361>.

[97] MARSLAND, S. *Machine learning: an algorithmic perspective*. Chapman and Hall/CRC, 2011. Disponível em: <https://doi.org/10.1201/9781420067194>.

[98] GOODFELLOW, I. et al. Generative adversarial nets. *Advances in neural information processing systems*, v. 27, 2014. Disponível em: <https://doi.org/10.48550/arXiv.1406.26617>.

[99] LI, Y. et al. The theoretical research of generative adversarial networks: an overview. *Neurocomputing*, Elsevier, v. 435, p. 26–41, 2021. Disponível em: <https://doi.org/10.1016/j.neucom.2020.12.114>.

[100] SALIMANS, T. et al. Improved techniques for training gans. *Advances in neural information processing systems*, v. 29, 2016. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2016/file/8a3363abe792db2d8761d6403605aeb7-Paper.pdf>.

[101] PAN, Z. et al. Recent progress on generative adversarial networks (gans): A survey. *IEEE Access*, IEEE, v. 7, p. 36322–36333, 2019. Disponível em: <https://doi.org/10.1109/ACCESS.2019.2905015>.

[102] JABBAR, A.; LI, X.; OMAR, B. A survey on generative adversarial networks: Variants, applications, and training. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 54, n. 8, oct 2021. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3463475>.

[103] KUMAR, M. P.; JAYAGOPAL, P. Generative adversarial networks: a survey on applications and challenges. *International Journal of Multimedia Information Retrieval*, Springer, v. 10, n. 1, p. 1–24, 2021. Disponível em: <https://doi.org/10.1007/s13735-020-00196-w>.

[104] WANG, K. et al. Generative adversarial networks: introduction and outlook. *IEEE/CAA Journal of Automatica Sinica*, IEEE, v. 4, n. 4, p. 588–598, 2017. Disponível em: <https://doi.org/10.1109/JAS.2017.7510583>.

[105] SAJEEDA, A.; HOSSAIN, B. M. Exploring generative adversarial networks and adversarial training. *International Journal of Cognitive Computing in Engineering*, Elsevier, 2022. Disponível em: <https://doi.org/10.1016/j.ijcce.2022.03.002>.

[106] BORJI, A. Pros and cons of gan evaluation measures. *Computer Vision and Image Understanding*, Elsevier, v. 179, p. 41–65, 2019. Disponível em: <https://doi.org/10.1016/j.cviu.2018.10.009>.

[107] LIU, Y. et al. Reconstruction of the meso-scale concrete model using a deep convolutional generative adversarial network (dcgan). *Construction and Building Materials*, v. 370, p. 130704, 2023. ISSN 0950-0618. Disponível em: <https://doi.org/10.1016/j.conbuildmat.2023.130704>.

[108] CHENG, J. et al. Generative adversarial networks: A literature review. *KSII Transactions on Internet and Information Systems (TIIS)*, Korean Society for Internet Information, v. 14, n. 12, p. 4625–4647, 2020. Disponível em: <https://doi.org/10.3837/tiis.2020.12.001>.

[109] XU, Q. et al. An empirical study on evaluation metrics of generative adversarial networks. *arXiv preprint arXiv:1806.07755*, 2018. Disponível em: <https://doi.org/10.48550/arXiv.1806.07755>.

[110] WAHEED, A. et al. Covidgan: data augmentation using auxiliary classifier gan for improved covid-19 detection. *Ieee Access*, IEEE, v. 8, p. 91916–91923, 2020. Disponível em: <http://dx.doi.org/10.1109/ACCESS.2020.2994762>.

[111] HUANG, S.; LEI, K. Igan-ids: An imbalanced generative adversarial network towards intrusion detection system in ad-hoc networks. *Ad Hoc Networks*, Elsevier, v. 105, p. 102177, 2020. Disponível em: <https://doi.org/10.1016/j.adhoc.2020.102177>.

[112] DING, H. et al. Imbalanced data classification: A knn and generative adversarial networks-based hybrid approach for intrusion detection. *Future Generation Computer Systems*, Elsevier, v. 131, p. 240–254, 2022. Disponível em: <https://doi.org/10.1016/j.future.2022.01.026>.

[113] LIU, X. et al. Nads-ra: network anomaly detection scheme based on feature representation and data augmentation. *IEEE Access*, IEEE, v. 8, p. 214781–214800, 2020. Disponível em: <https://doi.org/10.1109/ACCESS.2020.3040510>.

[114] DIN, N. U. et al. A novel gan-based network for unmasking of masked face. *IEEE Access*, IEEE, v. 8, p. 44276–44287, 2020. Disponível em: <https://doi.org/10.1109/ACCESS.2020.2977386>.

[115] ZHANG, H. et al. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 41, n. 8, p. 1947–1962, 2018. Disponível em: <https://doi.org/10.1109/TPAMI.2018.2856256>.

[116] ISLAM, M. J.; XIA, Y.; SATTAR, J. Fast underwater image enhancement for improved visual perception. *IEEE Robotics and Automation Letters*, IEEE, v. 5, n. 2, p. 3227–3234, 2020. Disponível em: <https://doi.org/10.1109/LRA.2020.2974710>.

[117] LEE, C.-K.; CHEON, Y.-J.; HWANG, W.-Y. Studies on the gan-based anomaly detection methods for the time series data. *IEEE Access*, IEEE, v. 9, p. 73201–73215, 2021. Disponível em: <https://doi.org/10.1109/ACCESS.2021.3078553>.

[118] SCHLEGL, T. et al. f-anogan: Fast unsupervised anomaly detection with generative adversarial networks. *Medical image analysis*, Elsevier, v. 54, p. 30–44, 2019. Disponível em: <https://doi.org/10.1016/j.media.2019.01.010>.

[119] DERHAB, A. et al. Intrusion detection system for internet of things based on temporal convolution neural network and efficient feature engineering. *Wireless Communications and Mobile Computing*, Hindawi, v. 2020, 2020. Disponível em: <https://doi.org/10.1155/2020/6689134>.

[120] SEO, E.; SONG, H. M.; KIM, H. K. Gids: Gan based intrusion detection system for in-vehicle network. In: IEEE. *2018 16th Annual Conference on Privacy, Security and Trust (PST)*. 2018. p. 1–6. Disponível em: <https://doi.org/10.1109/PST.2018.8514157>.

[121] ANDRESINI, G. et al. Gan augmentation to deal with imbalance in imaging-based intrusion detection. *Future Generation Computer Systems*, Elsevier, v. 123, p. 108–127, 2021. Disponível em: <https://doi.org/10.1016/j.future.2021.04.017>.

[122] ARAUJO-FILHO, P. F. de et al. Intrusion detection for cyber–physical systems using generative adversarial networks in fog environment. *IEEE Internet of Things Journal*, IEEE, v. 8, n. 8, p. 6247–6256, 2020. Disponível em: <https://doi.org/10.1109/JIOT.2020.3024800>.

[123] NIE, L. et al. Intrusion detection for secure social internet of things based on collaborative edge computing: a generative adversarial network-based approach. *IEEE Transactions on Computational Social Systems*, IEEE, v. 9, n. 1, p. 134–145, 2021. Disponível em: <https://doi.org/10.1109/TCSS.2021.3063538>.

[124] ZHANG, X.; WANG, J.; ZHU, S. Dual generative adversarial networks based unknown encryption ransomware attack detection. *IEEE Access*, IEEE, v. 10, p. 900–913, 2021. Disponível em: <https://doi.org/10.1109/ACCESS.2021.3128024>.

[125] SPEROTTO, A. et al. An overview of ip flow-based intrusion detection. *IEEE communications surveys & tutorials*, IEEE, v. 12, n. 3, p. 343–356, 2010. Disponível em: <https://doi.org/10.1109/SURV.2010.032210.00054>.

[126] SHARMA, R.; GULERIA, A.; SINGLA, R. An overview of flow-based anomaly detection. *International Journal of Communication Networks and Distributed Systems*, Inderscience Publishers (IEL), v. 21, n. 2, p. 220–240, 2018. Disponível em: <http://dx.doi.org/10.1504/IJCNDS.2018.10014505>.

[127] UMER, M. F.; SHER, M.; BI, Y. Flow-based intrusion detection: Techniques and challenges. *Computers & Security*, Elsevier, v. 70, p. 238–254, 2017. Disponível em: <https://doi.org/10.1016/j.cose.2017.05.009>.

[128] NDONDA, G. K.; SADRE, R. Network trace generation for flow-based ids evaluation in control and automation systems. *International Journal of Critical Infrastructure Protection*, Elsevier, v. 31, p. 100385, 2020. Disponível em: <https://doi.org/10.1016/j.ijcip.2020.100385>.

[129] JIRSIK, T. et al. Toward stream-based ip flow analysis. *IEEE Communications Magazine*, IEEE, v. 55, n. 7, p. 70–76, 2017. Disponível em: <https://doi.org/10.1109/MCOM.2017.1600972>.

[130] YANG, Z. et al. A systematic literature review of methods and datasets for anomaly-based network intrusion detection. *Computers & Security*, Elsevier, p. 102675, 2022. Disponível em: <https://doi.org/10.1016/j.cose.2022.102675>.

[131] RING, M. et al. A survey of network-based intrusion detection data sets. *Computers & Security*, Elsevier, v. 86, p. 147–167, 2019. Disponível em: <https://doi.org/10.1016/j.cose.2019.06.005>.

[132] GROUP, O. R. *Datasets used in Publications*. <http://www.uel.br/grupos/orion/datasets.html>, note = Accessed: 2023-04-10.

[133] NEIRA, A. B. de; KANTARCI, B.; NOGUEIRA, M. Distributed denial of service attack prediction: Challenges, open issues and opportunities. *Computer Networks*, Elsevier, p. 109553, 2023. Disponível em: <https://doi.org/10.1016/j.comnet.2022.109553>.

[134] ALI, T. E.; CHONG, Y.-W.; MANICKAM, S. Machine learning techniques to detect a ddos attack in sdn: A systematic review. *Applied Sciences*, MDPI, v. 13, n. 5, p. 3183, 2023. Disponível em: <https://doi.org/10.3390/app13053183>.

[135] SHANNON, C. E. A mathematical theory of communication. *The Bell system technical journal*, Nokia Bell Labs, v. 27, n. 3, p. 379–423, 1948. Disponível em: <https://doi.org/10.1002/j.1538-7305.1948.tb01338.x>.

[136] BHANJA, S.; DAS, A. Impact of data normalization on deep neural network for time series forecasting. *arXiv preprint arXiv:1812.05519*, 2018. Disponível em: <https://doi.org/10.48550/arXiv.1812.05519>.

[137] GÓMEZ-ESCALONILLA, V.; MARTÍNEZ-SANTOS, P.; MARTÍN-LOECHES, M. Preprocessing approaches in machine-learning-based groundwater potential mapping: an application to the koulikoro and bamako regions, mali. *Hydrology and Earth System Sciences*, Copernicus GmbH, v. 26, n. 2, p. 221–243, 2022. Disponível em: <https://doi.org/10.5194/hess-26-221-2022>.

[138] SCIKIT-LEARN. *User Guide*. <https://scikit-learn.org/stable/user_guide.html>. Accessed: 2023-04-06.

[139] RADFORD, A.; METZ, L.; CHINTALA, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. Disponível em: <https://doi.org/10.48550/arXiv.1511.06434>.

[140] KUMAR, G. Evaluation metrics for intrusion detection systems-a study. *Evaluation*, v. 2, n. 11, p. 11–7, 2014. ISSN 2321-8363.

[141] NASEER, S. et al. Enhanced network anomaly detection based on deep neural networks. *IEEE access*, IEEE, v. 6, p. 48231–48246, 2018. Disponível em: <https://doi.org/10.1109/ACCESS.2018.2863036>.

[142] ZAVRAK, S.; İSKEFIYELI, M. Anomaly-based intrusion detection from network flow features using variational autoencoder. *IEEE Access*, IEEE, v. 8, p. 108346–108358, 2020. Disponível em: <https://doi.org/10.1109/ACCESS.2020.3001350>.