Universidade Estadual de Londrina

VITOR GABRIEL DA SILVA RUFFO

DETECÇÃO E MITIGAÇÃO DE INTRUSÕES EM REDES DEFINIDAS POR SOFTWARE UTILIZANDO ANÁLISE DE FLUXOS IP E REDE ADVERSÁRIA GENERATIVA

LONDRINA
2025

VITOR GABRIEL DA SILVA RUFFO

DETECÇÃO E MITIGAÇÃO DE INTRUSÕES EM REDES DEFINIDAS POR SOFTWARE UTILIZANDO ANÁLISE DE FLUXOS IP E REDE ADVERSÁRIA GENERATIVA

Dissertação apresentada ao Programa de Mestrado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Mario Lemes Proença Jr.

Coorientador: Prof. Dr. Luiz Fernando Carvalho

LONDRINA

2025

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 e Mitigação de Intrusões em Redes Definidas por Software Utilizando Análise de Fluxos IP e Rede Adversária Generativa / Vitor Gabriel da Silva Ruffo. - Londrina, 2025.
80 f. : il.

Orientador: Mario Lemes Proença Jr..
Coorientador: Luiz Fernando Carvalho.
Dissertação (Mestrado em Ciência da Computação) - Universidade Estadual de Londrina, Centro de Ciências Exatas, Programa de Pós-Graduação em Ciência da Computação, 2025.
Inclui bibliografia.

1. Redes de Computadores - Tese. 2. Detecção de Intrusão - Tese. 3. Rede Adversária Generativa - Tese. I. Lemes Proença Jr., Mario. II. Carvalho, Luiz Fernando. III. Universidade Estadual de Londrina. Centro de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação. IV. Título.

CDU 519

VITOR GABRIEL DA SILVA RUFFO

DETECÇÃO E MITIGAÇÃO DE INTRUSÕES EM REDES DEFINIDAS POR SOFTWARE UTILIZANDO ANÁLISE DE FLUXOS IP E REDE ADVERSÁRIA GENERATIVA

Dissertação apresentada ao Programa de Mestrado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Mestre em Ciência da Computação.

BANCA EXAMINADORA

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

Prof. Dr. Wesley Attrot
Universidade Estadual de Londrina

Prof. Dr. Marcos Vinicius Oliveira de Assis
Universidade Federal do Paraná

Londrina, 15 de abril de 2025.

Este trabalho é dedicado aos meus pais e avós, pelo seu apoio absoluto e amor incondicional.

AGRADECIMENTOS

Aos meus pais e avós, por proverem seu amor, apoio e educação, permitindo que eu me desenvolva cada vez mais como pessoa.

Ao meu orientador, Dr. Mario Lemes Proença Jr., por sempre acreditar em mim e me apoiar ao longo da nossa parceria profissional de quase quatro anos.

Aos meus amigos do grupo de pesquisa Orion do departamento de computação da UEL, pela colaboração e troca de experiências que tornam a vida acadêmica mais produtiva e agradável.

Aos professores do curso de mestrado em ciência da computação da UEL, por impulsionarem ainda mais meu desenvolvimento pessoal após a graduação.

Aos servidores da UEL, por terem um papel fundamental no funcionamento da universidade e na formação de centenas de profissionais anualmente.

A CAPES e a Universidade Estadual de Londrina pela concessão de bolsa de estudos, possibilitando a execução desse trabalho e a realização do curso de mestrado.

"O crescimento e o conforto não coexistem." – Ginni Rometty

RUFFO, V. G. S.. **Detecção e Mitigação de Intrusões em Redes Definidas Por Software Utilizando Análise de Fluxos IP e Rede Adversária Generativa**. 2025. 80f. Dissertação (Mestrado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2025.

RESUMO

As redes de computadores facilitam tarefas do dia a dia, fornecendo serviços como *streaming* de dados, compras *online* e comunicação digital. Essas aplicações têm requerido cada vez mais capacidade e dinamicidade da rede para alcançar seus objetivos. As redes podem ser alvos de ataques e intrusões, comprometendo as aplicações e levando a potenciais perdas. Nesta dissertação, apresenta-se um sistema semi-supervisionado de detecção de anomalias de volume de tráfego para redes baseadas em fluxo IP. O sistema tem como principal componente uma Rede Adversária Generativa cuja arquitetura interna baseia-se na rede *1D-Convolutional Neural Network* (*1D-CNN*). O módulo de mitigação é acionado sempre que uma anomalia é detectada, bloqueando automaticamente os endereços IP suspeitos e promovendo o correto funcionamento da rede. Para fins de comparação, foram implementadas duas redes generativas que incorporam *Long Short-Term Memory* (*LSTM*) e *Temporal Convolutional Network* (*TCN*) na sua estrutura básica. Os experimentos são conduzidos em três bases de dados públicas: Orion, CIC-DDoS2019 e CIC-IDS2017. Os resultados sugerem que os três modelos de aprendizado profundo têm impactos distintos na rede generativa e, consequentemente, no desempenho geral do sistema de detecção. O sistema proposto implementado com *1D-CNN* mostrou-se superior aos outros modelos. Ele resolve o problema de *mode collapse*, é o mais eficiente em termos de custo computacional e alcança a segunda melhor pontuação para a tarefa de detecção de anomalias. O módulo de mitigação consegue descartar em média 96% dos fluxos IP anômalos, encaminhando a maioria do tráfego legítimo. As redes *1D-CNN*, *LSTM* e *TCN* também foram desenvolvidas separadamente da Rede Adversária Generativa para a condução de comparações de desempenho adicionais. O sistema proposto com base na rede generativa apresenta resultados superiores para as métricas de detecção de anomalias em comparação com esses modelos, obtendo um valor mínimo de *MCC* de 0,80.

Palavras-chave: Detecção de Intrusão. Mitigação de Ataques. Aprendizado Profundo. Rede Adversária Generativa. Rede Definida por Software.

RUFFO, V. G. S.. **Intrusion Detection and Mitigation on Software-Defined Networks Utilizing IP Flow Analysis and Generative Adversarial Network**. 2025. 80p. Master's Thesis (Master in Science in Computer Science) – State University of Londrina, Londrina, 2025.

ABSTRACT

Computer networks facilitate regular human tasks, providing services like data streaming, online shopping, and digital communications. These applications require more and more network capacity and dynamicity to accomplish their goals. The networks may be targeted by attacks and intrusions that compromise the applications that rely on them and lead to potential losses. We propose a semi-supervised detection system for traffic volume anomalies in IP flow-based networks. The system is implemented with a vanilla Generative Adversarial Network (GAN) whose internal architecture is based on the 1D-Convolutional Neural Network (1D-CNN). The mitigation module is triggered whenever an anomaly is detected, automatically blocking the suspect IPs and restoring the correct network functioning. We implemented two generative networks incorporating Long Short-Term Memory (LSTM) and Temporal Convolutional Network (TCN) into their internal structure for comparison purposes. The experiments are conducted on three public benchmark datasets: Orion, CIC-DDoS2019, and CIC-IDS2017. The results show that the three deep learning models considered have distinct impacts on the GAN model and, consequently, on the overall system performance. The 1D-CNN-based GAN implementation is the best since it reasonably solves the mode collapse problem, has the most efficient computational cost, and achieves competitive Matthews Correlation Coefficient scores for the anomaly detection task. Also, the mitigation module can drop an average of 96% of anomalous flows, forwarding the majority of legitimate traffic. We also implemented the 1D-CNN, LSTM, and TCN models separately from the *GAN*, aiming to conduct additional performance comparisons. The proposed system model shows improved overall results in the considered performance metrics compared to these models, reaching a minimum MCC value of 0.80.

Keywords: Intrusion Detection. Attack Mitigation. Deep Learning. Generative Adversarial Network. Software-Defined Network.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

1D-CNN	*1-Dimensional Convolutional Network*
1D-CNN-GAN	*1-Dimensional Convolutional Generative Adversarial Network*
t-SNE	*t-distributed stochastic neighbor embedding*
tanh	tangente hiperbólica
AE	*Autoencoder*
Adam	*Adaptive Moment Estimation*
AUROC	*Area Under the ROC curve*
BiGAN	*Bidirectional Generative Adversarial Network*
BiLSTM	*Bidirectional Long Short-Term Memory*
CNN	*Convolutional Neural Network*
CIC	*Canadian Institute for Cybersecurity*
DCGAN	*Deep Convolutional Generative Adversarial Network*
DDoS	*Distributed Denial of Service*
DL	*Deep Learning*
DoS	*Denial of Service*
DNN	*Deep Neural Network*
FPR	*False Positive Rate*
FP	Falso Positivo
FN	Falso Negativo
GAN	*Generative Adversarial Network*
GAN-GRU	*Generative Adversarial Network with Gated Recurrent Unit*
IA	Inteligência Artificial
IDS	*Intrusion Detection System*
IoT	*Internet of Things*

LIME	*Local Interpretable Model-Agnostic Explanations*
LSTM	*Long Short-Term Memory*
LSTM-GAN	*Long Short-Term Memory Generative Adversarial Network*
LR	*Logistic Regression*
LeakyReLU	*Leaky Rectified Linear Unit*
MCC	*Matthews Correlation Coefficient*
MAD-GAN	*Multivariate Anomaly Detection Generative Adversarial Network*
MQTT	*Message Queuing Telemetry Transport*
MLP	*Multi-Layer Perceptron*
NIDS	*Network Intrusion Detection System*
NaN	*Not a Number*
OCAN	*One-Class Adversarial Network*
PCA	*Principal Component Analysis*
RNN	*Recurrent Neural Network*
ROC	*Receiver Operating Characteristic*
ReLU	*Rectified Linear Unit*
RMSProp	*Root Mean Square Propagation*
RF	*Randon Forest*
RAM	*Random-Access Memory*
SDN	*Software-Defined Network*
SHAP	*SHapley Additive exPlanations*
SVM	*Support Vector Machine*
TCN	*Temporal Convolutional Network*
TCP/IP	*Transmission Control Protocol/Internet Protocol*
TCN-GAN	*Temporal Convolutinoal Generative Adversarial Network*
VP	Verdadeiro Positivo

VN	Verdadeiro Negativo
WGAN	*Wasserstein Generative Adversarial Network*
XAI	*eXplainable Artificial Intelligence*
XGBoost	*eXtreme Gradient Boosting*

LISTA DE SÍMBOLOS

p_r	Distribuição de probabilidades dos registros de tráfego legítimo
p_g	Distribuição de probabilidades dos registros sintetizados pelo gerador
p_z	Distribuição de probabilidades dos registros do espaço latente
G	Gerador
D	Discriminador
x	Registro de tráfego legítimo
x'	Registro sintetizado
z	Registro amostrado do espaço latente
k	Número de atualizações dos parâmetros do discriminador para uma única atualização do gerador
V	Função objetivo do modelo *GAN*
A	Função de custo otimizada pelo discriminador
B	Função de custo otimizada pelo discriminador e gerador
$\mathbb{E}$	Valor esperado (média)
σ	Função de ativação *sigmoid*
W	Pesos
b	Vieses
t	Tempo atual
f_t	Portão de lembrança de memórias passadas
C_{t-1}	Memórias passadas do neurônio *LSTM*
i_t	Portão de lembrança de memória atual
C'_t	Memória atual do neurônio *LSTM*
o_t	Portão de lembrança da memória final
C_t	Memória final do neurônio *LSTM*

h_t	Ativação do neurônio $LSTM$ para o tempo t
h_{t-1}	Ativação do neurônio $LSTM$ para o tempo $t-1$
κ	Tamanho de filtro de convoluções do modelo TCN
λ	Base de dilatação do modelo TCN
θ	Variável de dados
$H(x_\theta)$	Entropia de Shannon para a variável θ
υ	Valor não normalizado
υ'	Valor normalizado
min_θ	Valor mínimo de θ
max_θ	Valos máximo de θ
q	Tamanho da janela deslizante
d	Dimensionalidade do espaço latente
K	Número máximo de segundos que um certo endereço IP pode ser armazenado na lista segura ou lista de bloqueio
$O(n)$	Complexidade temporal linear
$O(1)$	Complexidade temporal constante

SUMÁRIO

1 INTRODUÇÃO

Atualmente, as redes de computadores são tão essenciais quanto serviços de energia e água para a sociedade. Redes de baixa latência e banda larga, como a *Ethernet* e a 5G, sobre a arquitetura *TCP/IP*, possibilitam a operação de diversas aplicações. Entre as mais comuns, tem-se comunicação a longas distâncias, computação em nuvem, *streaming* de áudio e vídeo, compras online, banco digital, redes sociais e IA assistente. O valor agregado pelas redes tem atraído cada vez mais usuários, aumentando o número de dispositivos conectados exponencialmente [1]. Consequentemente, a velocidade de conexão, volume de tráfego e quantidade de dados sensíveis armazenados têm aumentado drasticamente. Nesse contexto, expansão e adaptação se tornaram características necessárias para as redes poderem atender às demandas cada vez mais exigentes dos serviços concedidos aos seus usuários [2], [3], [4], [5].

Em uma arquitetura de redes tradicional, o plano de controle é acoplado ao plano de dados. O plano de controle é responsável por definir as regras de encaminhamento enquanto o plano de dados executa efetivamente a transmissão de dados. O hardware dos dispositivos que compõem a infraestrutura de rede integra mecanismos de ambos os planos. Cada um desses equipamentos é configurado individualmente utilizando a linguagem de programação especificada por cada vendedor [6]. Esse paradigma geralmente exige um alto custo de gerenciamento e atualização devido à natureza heterogênea da rede, dificultando a sua evolução para atender às demandas dinâmicas das aplicações [7]. Rede Definida por Software (*SDN*, do inglês *Software-Defined Network*) representa um paradigma que objetiva facilitar a gerência das redes, desacoplando o plano de dados do plano de controle. As *SDN*s apresentam maior flexibilidade a potenciais mudanças de requisitos [8], [9].

Outro elemento de gerência aplicado na manutenção de redes é a segurança cibernética [10]. Elas podem ser alvo de ataques cujo objetivo é o comprometimento da confidencialidade, integridade ou disponibilidade dos serviços de rede [11], [12], [13]. Esses serviços tornaram-se fundamentais para a sociedade e a sua falha pode levar a perdas monetárias, de reputação e até mesmo ameaçar vidas humanas [14], [15], [16]. Devido à evolução constante das redes e a criação de novos ataques, a comunidade científica investiga a área de segurança cibernética e de detecção de anomalias de rede por décadas. Os cientistas continuamente estudam soluções inovadoras para esse problema, como os Sistemas de Detecção de Intrusão de Redes (*NIDS*, do inglês *Network Intrusion Detection System*) [17], [18], [19], [20], [21].

NIDS são responsáveis por monitorar o tráfego de rede e identificar traços anormais nos dados. Quando uma anomalia é detectada, o sistema notifica os administradores de rede. Esses sistemas podem ser categorizados como baseados em assinatura ou baseados

em anomalia [22]. A segunda categoria tornou-se alvo de pesquisas por conseguir identificar anomalias conhecidas e desconhecidas [23], [24]. Um *NIDS* baseado em anomalias define um perfil de comportamento normal de tráfego (*baseline* de rede). Todo registro de tráfego que desvia suficientemente desse perfil é classificado como anômalo.

Ao longo dos anos, diversos métodos foram utilizados para implementar sistemas de detecção de intrusão [21], [25], [26], [27]. Modelos de Aprendizado Profundo (*DL*, do inglês *Deep Learning*) ganharam o foco dos pesquisadores por geralmente alcançarem os melhores resultados em comparação com outras soluções matemáticas [28]. *DL* é uma subárea de Aprendizado de Máquina formada por uma variedade de redes neurais. Esses modelos são treinados iterativamente para aprender a desempenhar tarefas específicas em certos tipos de dados. Exemplos incluem *Long Short-Term Memory* (*LSTM*), *1-Dimensional Convolutional Network* (*1D-CNN*) e *Temporal Convolutional Network* (*TCN*).

A Rede Adversária Generativa (*GAN*, do inglês *Generative Adversarial Network*) é um modelo de aprendizado profundo que tem sido amplamente estudado devido à sua capacidade de aprender a distribuição estatística de dados complexos [29]. Uma *GAN* tem duas redes internas que competem entre si durante o treinamento: o gerador e o discriminador [30]. Essas redes são treinadas para otimizar funções de custo que têm objetivos opostos. A tarefa do gerador é mapear vetores de ruído em registros de dados sintéticos. O discriminador é um classificador binário que aprende a discernir entre registros reais amostrados do conjunto de treinamento e registros gerados. Enquanto a rede geradora visa gerar exemplos de dados semelhantes aos reais, a rede discriminadora empenha-se em não ser enganada por dados sintéticos. Ao fim do treinamento, o gerador pode sintetizar registros que seguem a distribuição dos dados reais, tornando o discriminador incapaz de determinar a classificação correta de novos exemplos.

Nessa dissertação, propõe-se um sistema de detecção de anomalias de volume de tráfego em redes *TCP/IP* com suporte para coleta de fluxos IP, como as *SDN*s. A solução analisa fluxos de tráfego segundo a segundo, sendo composta por quatro módulos que processam os dados de maneira serial: i) coletor de tráfego; ii) pré-processador de dados; iii) detector de anomalias; e iv) mitigador. O primeiro módulo obtém os dados de todo o tráfego observado na rede no último segundo, enquanto o segundo módulo os prepara para a análise das etapas subsequentes. O detector de anomalias é responsável por comparar o padrão de volume de tráfego observado no segundo em análise com um perfil de comportamento normal esperado. Registros de tráfego são considerados como anômalos sempre que a diferença entre os seus padrões e o perfil de normalidade exceda um certo limiar de tolerância. Neste caso, os administradores são notificados e o módulo de mitigação é invocado para conter as anomalias e manter a rede em níveis normais de transmissão de dados. O perfil de normalidade é modelado utilizando uma rede adversária generativa baseada em convoluções unidimensionais (*1D-CNN-GAN*). O modelo *1D-CNN-GAN* é

treinado utilizando somente registros legítimos de tráfego para aprender a sua distribuição estatística p_r. A rede neural generativa foi escolhida para compor o sistema proposto dada a sua aplicabilidade na tarefa de definição de *baseline* de rede. Devido à sua característica de gerar dados sintéticos e modelar problemas, o modelo *GAN* tem sido objeto de estudo de inúmeros trabalhos de pesquisa [31], [32], [33].

As contribuições dessa dissertação são:

- Descrever um sistema semi-supervisionado para a detecção de anomalias em redes *TCP/IP*, contendo uma rede adversária generativa como componente principal;
- Apresentar um estudo sobre a aplicabilidade e desempenho do modelo *1D-CNN* na implementação do gerador e discriminador do modelo *GAN*;
- Detectar anomalias de volume de tráfego utilizando quatro variáveis de fluxo IP baseadas em entropia de Shannon;
- Empregar o método de redução de dimensionalidade *t-SNE* para visualizar os padrões numéricos das variáveis de entropia em duas dimensões;
- Um algoritmo de mitigação para conter automaticamente as anomalias detectadas, promovendo a confidencialidade, integridade e disponibilidade de rede sem intervenção humana;
- Conduzir experimentos em três bases de dados públicas: Orion [34], CIC-DDoS2019 [35] e CIC-IDS2017 [36];
- Comparar o modelo *1D-CNN-GAN* proposto com outros modelos que utilizam redes neurais profundas: *LSTM-GAN, LSTM, 1D-CNN, TCN-GAN* e *TCN*;
- Aplicar a técnica de inteligência artificial explicável *SHAP* para investigar como as variáveis de entropia de entrada influenciam a saída do modelo *1D-CNN-GAN*.

O restante desta dissertação está organizada da seguinte forma: o capítulo 2 apresenta conceitos fundamentais; o capítulo 3 discute os trabalhos relacionados; o capítulo 4 descreve o sistema de detecção proposto; o capítulo 5 expõe os cenários experimentais e os seus respectivos resultados; o capítulo 6 elabora as conclusões.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Rede Definida por Software

As Redes Definidas por Software são redes baseadas em fluxo IP que simplificam a manutenibilidade e promovem adaptabilidade. Nesse paradigma, o plano de controle é desacoplado do plano de dados. Os equipamentos que formam o plano de dados, como *switches* e roteadores, são somente responsáveis pela transmissão de pacotes. Esses dispositivos comumente possuem tabelas populadas por regras que definem como os dados devem ser encaminhados. O controlador *SDN* representa o plano de controle e possuí visão global da rede, sendo um elemento logicamente centralizado. Esse equipamento instala a lógica de controle nas tabelas dos dispositivos de encaminhamento a partir da interface *southbound* [37]. O controlador também age como um sistema operacional de rede. Ele abstrai a complexidade da infraestrutura e provê uma interface *northbound* de alto nível que permite a programabilidade da rede. O administrador pode, então, especificar o comportamento, políticas e soluções de gerência desejadas mediante softwares que executam no plano de aplicação [38].

O paradigma *SDN* traz diversas vantagens para a administração de redes, porém ele também pode ser alvo de ataques. Por exemplo, a centralização de inteligência no controlador cria um ponto central de falha. Essa vulnerabilidade pode ser explorada por ataques de negação de serviço distribuídos (*DDoS*, do inglês *Distributed Denial of Service*) [8]. As *SDN*s também podem ser alvos de outros tipos de ataques, como escaneamento, *spoofing, hikacking, tampering* e *man-in-the-middle* [6].

2.2 Detecção de Anomalias e Intrusões de Redes

Ataques e anomalias de tráfego podem comprometer o funcionamento de serviços de rede, potencialmente causando perdas aos administradores e usuários. Um exemplo comum é o ataque *DDoS*. Nesse ataque, o agente malicioso controla uma rede de máquinas infectadas por *malware*, visando encaminhar mais tráfego do que um servidor ou rede alvo pode processar. A vítima tem seus recursos exauridos, comprometendo a disponibilidade dos seus serviços para usuários legítimos [39], [40], [41]. Os ataques e intrusões têm evoluído e se adaptado ao longo dos anos, reforçando a importância da área de detecção de anomalias [42], [43], [44]. Os sistemas de detecção de intrusão de redes representam uma solução amplamente estudada e difundida entre a comunidade científica. O seu objetivo é preservar a confidencialidade, integridade e disponibilidade dos serviços de rede [27]. Um *NIDS* regularmente coleta e analisa o tráfego de rede, alertando os administradores quando traços anômalos são encontrados nos dados.

Os *NIDS* podem ser classificados como baseados em assinatura ou baseados em anomalias em função da sua estratégia de detecção. O primeiro tipo de sistema mantém um banco de dados composto por assinaturas de ataques previamente mapeados. Ele monitora o tráfego, verificando se os registros em análise possuem os padrões de alguma assinatura armazenada. Alarmes são gerados para cada padrão encontrado nos dados. Esse método tende a ter uma baixa taxa de falsos alarmes, porém ele pode detectar somente anomalias conhecidas.

Sistemas baseados em anomalias podem detectar anomalias desconhecidas. Esse método modela o comportamento de tráfego legítimo com bases no processamento de dados históricos [45]. O comportamento dos registros de tráfego em análise é calculado e comparado com o modelo de normalidade previamente estabelecido. Caso a sua diferença exceda um limiar de tolerância, o sistema gera um alarme indicando a ocorrência de uma anomalia. Um ponto negativo dessa abordagem é o potencial aumento na taxa de falsos alarmes, pois variações de comportamento de tráfego benigno podem ser interpretadas como uma anomalia. Comumente, o perfil de comportamento de rede evolui ao longo do tempo, requerendo atualizações recorrentes do modelo de normalidade [27].

2.3 Rede Adversária Generativa

A Rede Adversária Generativa é um modelo de aprendizado profundo introduzido em 2014 por Goodfellow *et al.* [29]. A Figura 1 ilustra a arquitetura da *GAN*, sendo composta de duas redes com funções objetivos opostas: o gerador G e o discriminador D. O gerador visa criar registros sintéticos semelhantes aos exemplos reais de treinamento. Ele aprende a mapear vetores de ruído aleatório z amostrados da distribuição p_z do espaço latente para registros $x' = G(z)$. A distribuição dos dados sintetizados é referenciada como p_g. Ao longo do treinamento, a rede geradora aproxima p_g da distribuição dos dados reais p_r. O discriminador busca não ser enganado pelos exemplos gerados. Ele atua como um classificador binário treinado para distinguir entre dados sintéticos $x' = G(z) \sim p_g$ e dados reais $x \sim p_r$ [46].

O modelo *GAN* é treinado com base no algoritmo de gradiente descendente [47]. Para um número finito de iterações (épocas), o conjunto de dados é embaralhado e dividido em lotes. Para cada lote, ajustam-se os parâmetros internos da rede discriminadora k vezes e os do gerador apenas uma vez. Esses ajustes ocorrem de maneira separada para cada rede. Enquanto os parâmetros do discriminador são modificados, os do gerador permanecem inalterados, e vice-versa.

Os parâmetros são ajustados para otimizar a função objetivo V definida nas equações (2.1), (2.2) e (2.3), onde $\mathbb{E}$ representa o valor esperado [48]. O objetivo do gerador é minimizar V ao reduzir a função B, induzindo o discriminador a classificar as suas entra-

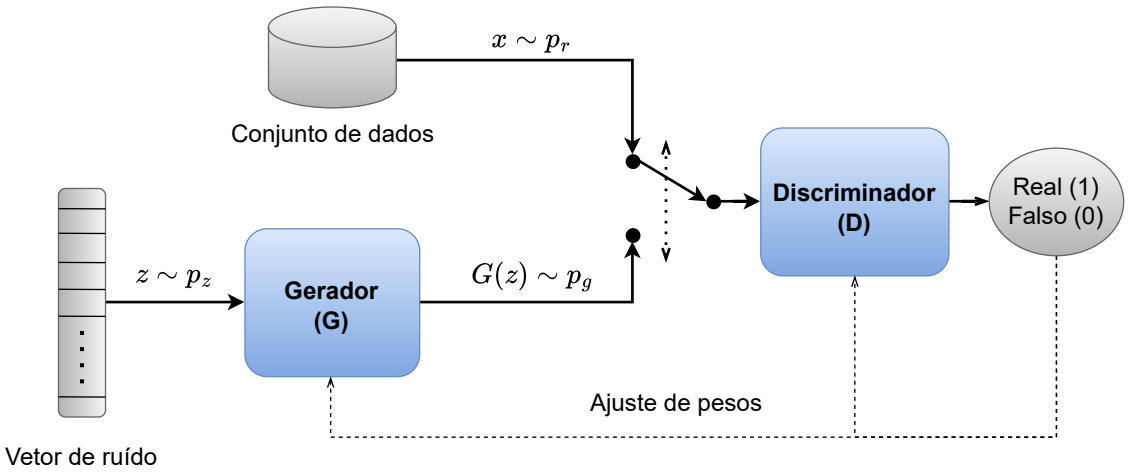

Figura 1 – Rede Adversária Generativa.

das erroneamente. A saída da função B é reduzida quando o valor esperado de $D(G(z))$ se aproxima de 1, levando o logaritmo de $1 - D(G(z))$ para infinito negativo. Por outro lado, o propósito do discriminador é maximizar a função V aumentado o valor das funções A e B, de modo a classificar as entradas corretamente [49]. A função A aumenta de valor quando o $\mathbb{E}$ de $D(x)$ atinge 1 e, consequentemente, o seu logaritmo tende a 0. A função B tem o seu valor aumentado assim que $\mathbb{E}$ de $D(G(z))$ é reduzido para 0, tornando o logaritmo de $1 - D(G(z))$ próximo de 0.

$$\min_G \max_D V(D, G) = A(D) + B(D, G) \tag{2.1}$$

$$A(D) = \mathbb{E}_{x \sim p_r(x)}[\log D(x)] \tag{2.2}$$

$$B(D, G) = \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.3}$$

O objetivo geral do treinamento do modelo *GAN* é convergir para o Equilíbrio de Nash [29]. Esse estado é alcançado quando os parâmetros internos de ambas as redes representam o ponto de mínimo ou máximo global das suas respectivas funções objetivo. Nenhuma mudança a qualquer parâmetro pode otimizar o valor dessas funções ainda mais. Nesta etapa do treinamento, o gerador aprende efetivamente a distribuição dos dados reais ($p_g \approx p_r$). Os exemplos de dados sintéticos enganam completamente o discriminador. Ou seja, a rede discriminadora não é mais capaz de diferenciá-los dos exemplos reais. Ela externaliza a sua incerteza ao produzir valores de saída próximos de 0,5 para qualquer entrada processada.

A comunidade científica identificou falhas determinantes na proposta original da Rede Adversária Generativa, como *mode collapse*, instabilidade de treinamento e dissipação de gradientes. O primeiro problema indica a falta de diversidade nos exemplos sintetizados pelo gerador. O segundo refere-se à dificuldade de alcançar o Equilíbrio de Nash, uma vez que o treinamento da *GAN* é naturalmente instável. Nessa condição, o discriminador tende a classificar as suas entradas corretamente, minimizando os gradientes enviados ao gerador e prevenindo a sua evolução. Diversos autores propuseram implementações alternativas para o modelo *GAN* visando resolver os problemas inerentes à rede original. Por exemplo, a *GAN* convolucional profunda, *GAN* condicional e *GAN* de Wasserstein [50].

2.4 Rede Neural Convolucional

A Rede Neural Convolucional (*CNN*, do inglês *Convolutional Neural Network*) é um modelo de aprendizado profundo que foi inicialmente proposto para o processamento de imagens [51]. Os modelos *CNN* também apresentam desempenho notável para outros tipos de tarefa, como predição de séries temporais e análise de linguagem natural. Essa rede utiliza menos parâmetros internos do que uma rede neural convencional por estabelecer menos conexões entre os seus neurônios. Ela é composta de quatro tipos de camadas, como ilustrado na Figura 2: convolucional, *pooling*, *flatten* e densa [52].

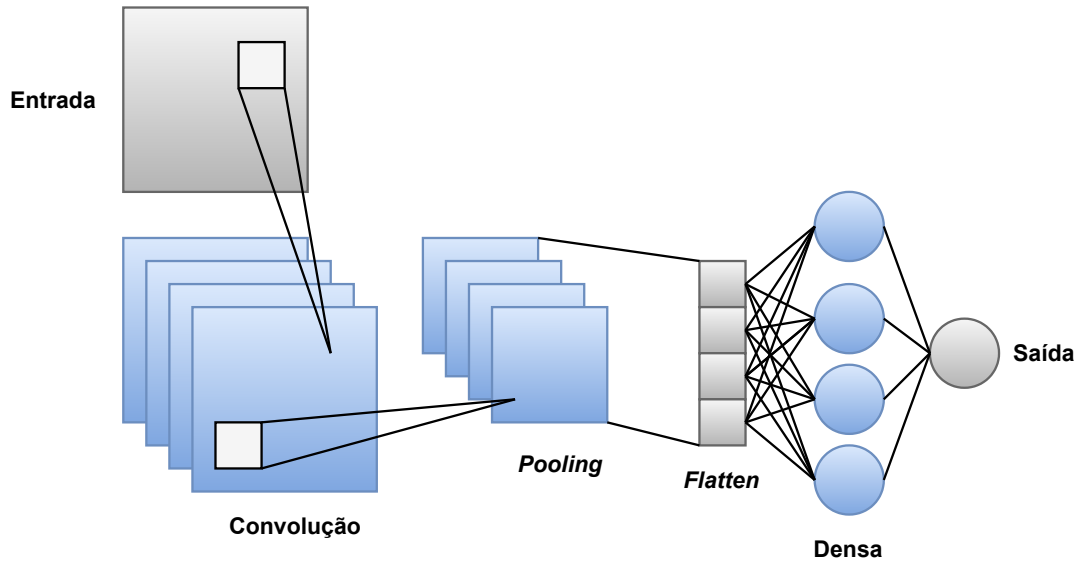

Figura 2 – Rede Neural Convolucional.

As camadas convolucionais extraem matrizes de padrões complexos dos dados de entrada mediante operações de convolução, reduzindo a sua dimensionalidade no processo. Uma operação de convolução é executada iterando os dados e multiplicando-os por uma matriz de pesos que comumente tem tamanho inferior à entrada. As camadas de *pooling*

agem como uma segunda etapa de redução de dimensionalidade, comprimindo os dados. Comumente, múltiplas camadas convolucionais e de *pooling* são concatenadas para calcular as características essenciais dos dados [53]. A camada *flatten* é aplicada na sequência para converter as matrizes de características em formato de vetor, adequado para o processamento da camada seguinte. Por fim, as camadas densas são responsáveis por mapear o vetor de informações extraídas para o formato correto de saída. Por exemplo, em uma tarefa de classificação de imagens, a saída poderia ser representada por uma camada densa de n neurônios, onde n indica o número possível de classes de imagens.

2.5 *Long Short-Term Memory*

A rede *Long Short-Term Memory* é uma Rede Neural Recorrente (*RNN*, do inglês *Recurrent Neural Network*) proposta em 1997 por Hochreiter *et al.* [54]. Essa rede é tipicamente aplicada em tarefas de modelagem de sequência. O modelo *LSTM* é composto por neurônios especiais que possuem retroalimentação, armazenando memória de dados processados previamente. Cada vez que um exemplo de dados é alimentado na rede, o resultado do processamento de instâncias anteriores são considerados na análise atual [55]. Devido à sua retenção de memória, as redes *LSTM* podem aprender correlações de longo prazo em dados sequenciais.

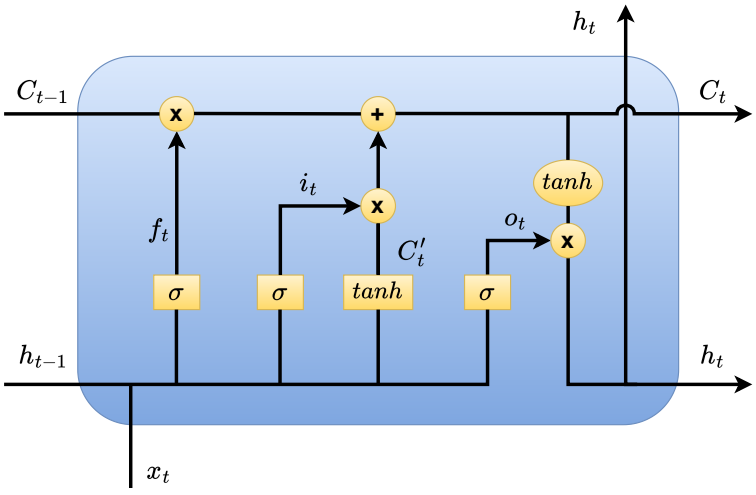

Figura 3 – *Long Short-Term Memory.*

É ilustrado na Figura 3 uma representação de um neurônio *LSTM*. Os seus passos de processamento são descritos pelas equações (2.4) a (2.9) [56]. Os símbolos σ e *tanh* representam as funções *sigmoid* e tangente hiperbólica, respectivamente. Um portão é uma rede neural interna que gerencia a memória do neurônio. Cada portão tem um conjunto exclusivo de pesos W e vieses b, usados para calcular uma combinação não linear de sua entrada. O portão f_t (2.4) é um fator que determina o quanto da memória passada do

neurônio C_{t-1} deve ser lembrada. O portão i_t (2.5) decide o quanto manter da memória C'_t (2.6) calculada no tempo de análise atual. C_t (2.7) representa a memória atualizada do neurônio dada pela soma das frações da memória passada e da memória atual. A ativação do neurônio é obtida a partir de h_t (2.9) como sendo uma função da memória atualizada do neurônio fracionada pelo portão o_t (2.8).

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{2.4}$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2.5}$$

$$C'_t = tanh(W_C[h_{t-1}, x_t] + b_C) \tag{2.6}$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot C'_t \tag{2.7}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.8}$$

$$h_t = o_t \cdot tanh(C_t) \tag{2.9}$$

2.6 Rede Convolucional Temporal

A Rede Convolucional Temporal é um modelo de aprendizado profundo formalizado em 2018 por Bai *et al.* [57]. Essa rede é estudada como uma alternativa às Redes Neurais Recorrentes para a tarefa de modelagem de sequência. A Figura 4 mostra o elemento que compõe modelos *TCN*, o bloco residual. Esse elemento consiste em duas camadas de convolução causal dilatada. Ambas são seguidas de funções de normalização de pesos, ativação e regularização para evitar a explosão de gradientes, adicionar não linearidade e prevenir *overfitting*, respectivamente. As operações de convolução são dilatadas, pois os seus filtros saltam certos dados de entrada com base em um fator de dilatação, aumentando o alcance em termos de tempo. Essas convoluções são causais, já que manipulam apenas elementos posicionados no tempo atual (t) ou em tempos passados ($t-1$). Um modelo *TCN* tipicamente empilha múltiplos blocos residuais para extrair padrões de alto nível dos dados de entrada e produzir uma predição de saída.

O campo receptivo de uma *TCN* é o tamanho máximo da entrada que um filtro da segunda convolução do bloco residual no topo da pilha pode alcançar e processar. Esse campo de alcance deve ser pelo menos do tamanho da sequência de entrada para garantir que o modelo possa considerar toda a série temporal em seus cálculos. Se esse

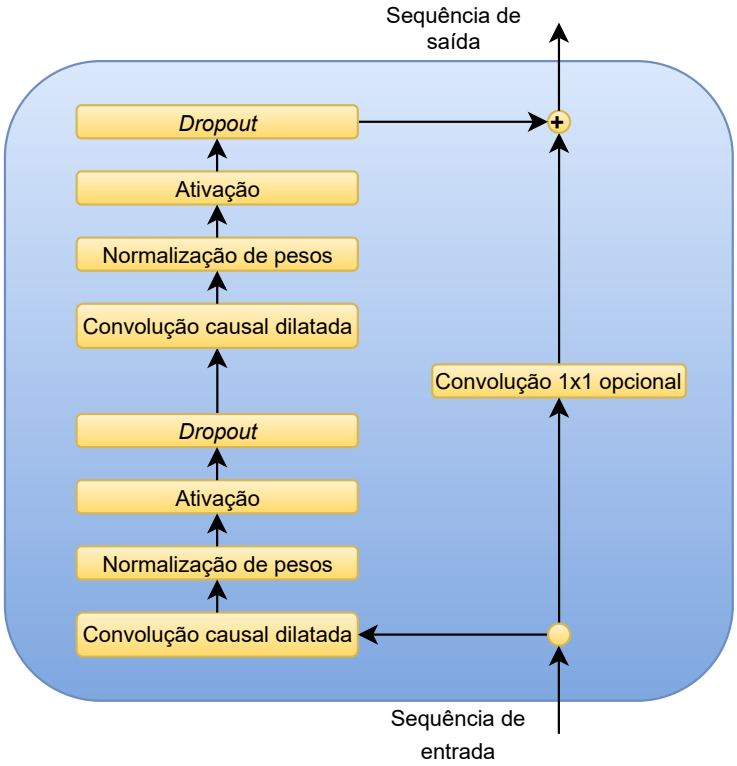

Figura 4 – Bloco Residual do modelo *TCN*.

critério for atendido, o último elemento da sequência de saída produzida pelo modelo *TCN* representa uma função de toda a sequência de entrada. Essa característica torna a *TCN* um modelo competitivo entre as redes recorrentes. Ela pode atribuir o mesmo peso a todos os elementos da sequência de entrada, independente de sua posição no tempo.

O campo receptivo é definido por (2.10) [58]. Ele pode ser expandido ao aumentar o tamanho de filtro das convoluções κ, a base de dilatação λ ou o número de blocos residuais empilhados n. Os fatores de dilatação λ^i de cada camada convolucional dependem do nível i que seu bloco está na pilha de blocos residuais da *TCN*. Um par de convoluções de um mesmo bloco têm fatores de dilatação iguais. Todas as convoluções têm o mesmo tamanho de filtro, independente do nível de seu bloco.

$$Campo\ receptivo = 1 + 2(\kappa - 1)\sum_{i=0}^{n-1}\lambda^i \tag{2.10}$$

Dependendo do tamanho da sequência de entrada, o campo receptivo necessário para cobri-la completamente pode fazer o modelo *TCN* ficar complexo em termos de quantidade de blocos residuais. Esse aumento de profundidade em redes neurais comumente causa problemas como a dissipação de gradientes durante o treinamento. A arquitetura da *TCN* evita esse problema adicionando conexões residuais em seus blocos, permitindo que os gradientes fluam através das múltiplas camadas convolucionais [59]. Como apre-

sentado no lado direito da Figura 4, esse mecanismo, se necessário, equaliza as dimensões de entrada e saída usando uma operação de convolução 1×1 e adiciona a entrada à saída do bloco.

2.7 Inteligência Artificial Explicável

Modelos de aprendizado de máquina são aplicados para resolver problemas em diversas áreas de pesquisa. Comumente, quanto maior a complexidade computacional de um modelo, melhor é o seu desempenho no processamento de dados e realização de inferências. Esse aumento no desempenho está atrelado à redução da transparência dos modelos, tornando-os incapazes de prover explicações sobre seu funcionamento interno e a lógica por trás de suas decisões [60]. Por exemplo, árvore de decisão, regressão linear e regressão logística são exemplos clássicos de aprendizado de máquina que são mais simples e compreensíveis. Esses modelos comumente têm desempenho inferior e não são viáveis para processar dados complexos reais. Por outro lado, as redes neurais profundas, como *DNN, CNN* e *RNN*, alcançam desempenhos cada vez mais altos em detrimento de sua interpretação e entendimento do seu processo de inferência. Esses modelos são comumente conhecidos como "caixas pretas" devido à sua falta de transparência operacional [61].

Aplicações que lidam com dados sensíveis requerem que os modelos de aprendizado alcancem alto desempenho e que também sejam transparentes e compreensíveis. A transparência é importante para agregar confiança aos resultados produzidos pelos modelos, viabilizando o seu uso em cenários críticos. Exemplos de aplicações críticas incluem sistemas autônomos, soluções bancárias, diagnósticos de doenças e algoritmos militares [62]. Neste contexto, a área de pesquisa em inteligência artificial explicável (*XAI*, do inglês *eXplainable Artificial Intelligence*) estuda técnicas para prover transparência e confiabilidade a modelos de aprendizado profundo. O objetivo é alcançar uma troca (*trade-off*) entre compreensão e desempenho do modelo para que as redes neurais profundas sejam aplicáveis em cenários críticos [42], [63].

As características de compreensividade e transparência de modelos "caixa preta" são alcançadas por meio de explicabilidade e interpretabilidade provenientes da aplicação de técnicas de *XAI* [64]. A explicabilidade refere-se à explicação da lógica e mecanismos por trás das decisões tomadas pelo modelo em análise. Ela responde o porquê de uma certa inferência ter sido produzida a partir de uma dada entrada, permitindo que usuários finais possam verificar se o modelo toma decisões corretas e não enviesadas com base nos dados apresentados. A interpretabilidade diz respeito ao fornecimento de meios para que usuários com o conhecimento técnico adequado compreendam o funcionamento interno da arquitetura dos modelos "caixa preta". Ela vai responder como ocorre o processo de geração de inferências do modelo.

Técnicas de *XAI* podem ser categorizadas conforme o seu i) escopo de dados; ii) tipo de modelo considerado; iii) e estágio de aplicação [65]. Abaixo descreve-se cada uma dessas categorias:

- i) Escopo: técnicas de explicabilidade podem ser classificadas como locais ou globais. As locais fornecem explicações sobre uma única predição do modelo em análise, enquanto as globais revelam o seu comportamento de maneira geral para um conjunto de dados completo.
- ii) Modelo: algumas técnicas são desenvolvidas para explicar modelos específicos (*model-specific*). Já outras são genéricas e podem ser utilizadas para compreender qualquer tipo de modelo (*model-agnostic*).
- iii) Estágio: essas técnicas são aplicadas em um dos três estágios de desenvolvimento de um modelo "caixa preta". Isto é, são executadas antes do treinamento (*ante-hoc*), durante o treinamento (*intrinsic*) ou após o treinamento (*post-hoc*).

O *SHAP* (*SHapley Additive exPlanations*) é um exemplo de técnica de *XAI* para agregar explicabilidade medindo a contribuição (valor *SHAP*) de cada variável de entrada na saída produzida por um modelo [66]. Esse método pode ser aplicado tanto de maneira local ou global aos dados. Ele também é independente do modelo em análise (*model-agnostic*), além de ser aplicado após o treinamento da caixa preta (*post-hoc*). O *SHAP* baseia-se na teoria de jogos cooperativos, onde se têm diversos jogadores e cada um deles apresenta uma contribuição particular para o resultado do jogo. Neste cenário, os atributos de entrada de um modelo caixa preta representam os jogadores e a saída do modelo indica o resultado do jogo [67]. Na seção 5.3.2, apresentam-se experimentos onde a técnica *SHAP* é aplicada de maneira global aos dados para gerar explicações da relação entre a entrada e a saída do modelo proposto nesta dissertação.

3 TRABALHOS RELACIONADOS

Detecção de anomalias e intrusões é uma área de pesquisa que tem estado ativa por muitos anos devido à constante evolução das redes e sofisticação de ataques. Levantamentos recentes apontam diversos problemas em aberto, como por exemplo, disponibilidade de bases de dados de qualidade, integração de *XAI* em *NIDS*, complexidade computacional elevada, escalabilidade, entre outros. Diversos trabalhos propõem diferentes maneiras de aproximar uma solução para a detecção de anomalias [68], [69], [70], [71].

3.1 Detecção de Intrusão de Redes baseada em Aprendizado Profundo

Recentemente, a comunidade científica tem explorado modelos de aprendizado profundo para resolver o problema de detecção de anomalias e intrusões de rede. Por exemplo, Cherian et al. [72], Tayfour et al. [73] e Soltani et al. [74] apresentaram sistemas de detecção de intrusão que aplicam a rede *LSTM* para extrair padrões temporais do tráfego de rede. Essas características de alto nível são encaminhadas para camadas densas, que determinam o rótulo adequado para o tráfego correspondente.

Hnamte *et al.* [56] desenvolveram um sistema de detecção de intrusão implementado com uma combinação de *CNN*, *LSTM* bidirecional e *DNN*. O sistema utiliza operações de convolução para extrair padrões de alto nível dos dados, seguido de uma camada de *LSTM* bidirecional para realizar modelagem de sequência. Camadas densas processam as informações identificadas pelas camadas anteriores para calcular a classificação do registro de entrada. Os autores selecionaram as bases CIC-IDS2018 e Edge_IIoT para conduzir os experimentos. O sistema proposto apresentou acurácia superior em comparação com modelos consolidados como *DNN, CNN, Autoencoder* e *LSTM*.

Houda *et al.* [75] discutiram um *framework* explicável para detecção de intrusões em redes IoT, integrando técnicas de *XAI*, como *SHAP, LIME* e *RuleFit* a um modelo *DNN*. O objetivo dos autores é prover uma interface que forneça transparência sobre as decisões da rede neural profunda para que seus usuários possam confiar e utilizar os resultados do sistema. Os experimentos são conduzidos nas bases de dados NSL-KDD e UNSW-NB15. As técnicas *RuleFit* e *SHAP* mostram que poucos atributos têm real impacto nas respostas produzidas pelo modelo, enquanto a maioria não auxilia na discriminação das classes de tráfego. O sistema proposto supera trabalhos similares em termos da pontuação *F1-score*, além de fornecer explicabilidade sobre as decisões do modelo *DNN*.

Esta dissertação é focada em redes baseadas em fluxo IP, como as Redes Definidas por Software. O paradigma *SDN* simplifica a tarefa de gerência, facilitando a inclusão

de novas funções na rede, como sistemas de detecção de intrusão [8]. Alguns autores constroem o seu *NIDS* para ser instalado especificamente em *SDN*s. Shaji *et al.* [76] propuseram uma ferramenta para detectar ataques *DDoS* em ambientes *SDN*. O sistema é implementado com uma rede neural *Multi-layer Perceptron* (*MLP*) que modela o problema como uma tarefa de classificação multi-classe. Além de detectar ataques *DDoS*, o modelo também os classifica. Uma base de dados de uma *SDN* emulada é utilizada para treinar e testar o modelo, alcançando uma acurácia de 98,81% com uma taxa de falsos positivos de 0,02%. A proposta também apresenta um *F1-score* superior a de dois trabalhos similares.

Kumar *et al.* [77] descreveram uma nova técnica de seleção de variáveis de entrada para melhorar o desempenho de classificadores de intrusão em Redes Definidas por Software. Essa seleção é baseada no algoritmo *whale optimization* em combinação com a pontuação de Fisher e ganho de informação. Os autores avaliaram a solução em combinação com classificadores como *SVM* e *LR*. O sistema é testado utilizando as bases de dados InSDN e CIC-IDS2017 e comparado com trabalhos do estado da arte. Prova-se que o método de seleção proposto pode melhorar a acurácia geral de detecção de intrusão.

3.2 Detecção de Intrusão de Redes baseada em Rede Adversária Generativa

A Rede Adversária Generativa é um modelo de aprendizado profundo que tem ganhado atenção dos cientistas devido à sua capacidade de reconhecimento de padrões. Diversos autores têm utilizado *GAN* para implementar sistemas de detecção de intrusão inovadores. A Figura 5 apresenta uma taxonomia de como os trabalhos integram o modelo *GAN* aos seus *NIDS*. Uma aplicação comum é a geração de exemplos de dados sintéticos para resolver o problema de desbalanceamento de classes. O modelo também pode ser aplicado para implementar o próprio algoritmo de detecção. Neste caso, o sistema pode ainda ser classificado como baseado em gerador ou baseado em discriminador. O primeiro utiliza o erro de reconstrução de exemplos, enquanto o segundo recorre à saída do discriminador para detectar anomalias. Alguns autores ainda implementam uma solução híbrida que aproveita tanto o gerador quanto o discriminador no processo de detecção.

Desbalanceamento de classes é um problema comum encontrado em bases de dados para a avaliação de sistemas de detecção de intrusão. Ele representa a distribuição não uniforme de classes, ou rótulos, para os registros de dados. Esse desbalanceamento reduz o desempenho de modelos de aprendizado de máquina supervisionados, que dependem de dados rotulados para o seu treinamento [78]. Park *et al.* [79] propuseram uma solução para esse problema em seu trabalho. Eles introduziram um sistema de detecção que combina diversas arquiteturas de aprendizado profundo. Os dados de entrada são pré-processados e usados para treinar uma *GAN*. Esse modelo é utilizado para gerar novos registros e

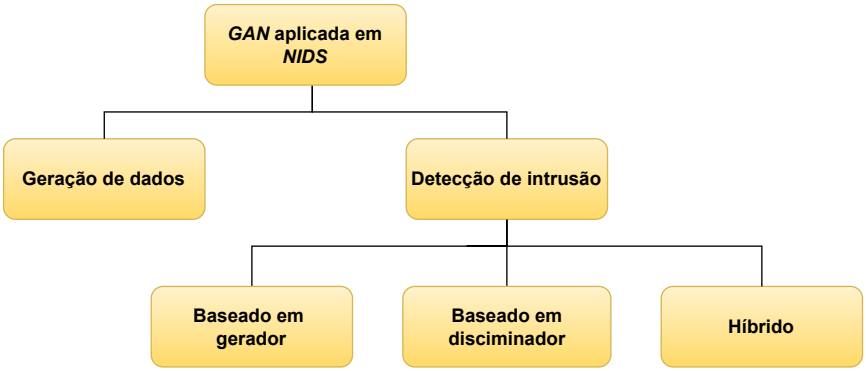

Figura 5 – Taxonomia da integração de *GAN* em sistemas de detecção de intrusão de redes.

balancear as classes da base de dados. Uma rede autoencoder é treinada no novo conjunto de dados para realizar a extração de padrões. O último componente do sistema aplica um classificador como *DNN, CNN* ou *LSTM* para realizar a identificação de diferentes tipos de tráfego. A proposta é testada nas bases NSL-KDD e UNSW-NB15. Prova-se experimentalmente que ela supera modelos básicos como *DNN, CNN, LSTM* e modelos híbridos que combinam *AE* com *DNN* e *CNN*.

Kumar *et al.* [80] descreveram um modelo baseado em *GAN* para gerar dados de classes de ataque pouco representadas e melhorar o seu desempenho de detecção de intrusões. O sistema usa um *Autoencoder* para calcular as variáveis mais relevantes do tráfego de entrada e uma *GAN* condicional de Wasserstein para gerar dados extras. Um classificador *XGBoost* é treinado na base de dados expandida utilizando as variáveis mais relevantes para distinguir entre tráfego legítimo e anômalo. A proposta foi testada nas bases NSL-KDD, UNSW-NB15 e BoT-IoT, produzindo resultados melhores do que um trabalho similar da literatura.

A rotulagem de dados é uma tarefa cara e complexa em ambientes de redes reais [81]. Para evitar esse problema, muitos trabalhos implementam *NIDS* não supervisionados ou semi-supervisionados, desconsiderando possíveis desbalanceamentos de classes. Nesses casos, o modelo *GAN* é comumente utilizado na implementação do próprio algoritmo de detecção. Alguns sistemas detectam anomalias aplicando o gerador para calcular o erro de reconstrução de registros de tráfego. Boppana *et al.* [82] desenvolveram um sistema para a detecção de intrusões em redes MQTT-IoT que se baseia em *Autoencoder* e rede adversária generativa. Os modelos são treinados de modo não supervisionado para reconstruir tráfego benigno com precisão, aprendendo a distribuição desses dados. Anomalias são detectadas quando o erro de reconstrução de um registro ultrapassa um limiar pré-definido. O sistema apresentou resultados superiores a algoritmos clássicos, como *Autoencoder*, *SVM* e *Isolation Forest* para a base de dados MQTT-IoT-IDS2020.

Yao *et al.* [83] descrevem um *NIDS* não supervisionado para promover a segurança de redes *IoT*. A solução emprega a rede adversária generativa bidirecional para calcular erro de reconstrução e detectar anomalias. Registros de tráfego benigno tendem a ter um erro pequeno, enquanto exemplos malignos são reconstruídos com baixa precisão, obtendo um erro maior. Esse sistema é instalado em ambientes de *fog computing* para solucionar problemas de escalabilidade, complexidade e latência. Os autores avaliaram o desempenho da sua proposta nas bases UNSW-NB15 e CIC-IDS2017, que obteve resultados promissores em comparação com outros trabalhos da comunidade científica.

A saída produzida pela rede discriminadora também pode ser utilizada no algoritmo de detecção baseado em *GAN*. Li *et al.* [84] descrevem um mecanismo de classificação de tráfego utilizando uma nova técnica de extração de padrões e um modelo híbrido de *autoencoder* e *GAN*. O módulo de extração calcula os padrões intrínsecos dos dados e os encaminha para o modelo *autoencoder-GAN*, que utiliza a saída do discriminador e um limiar para classificar esse tráfego como benigno ou maligno. A proposta é avaliada nas bases NSL-KDD e UNSW-NB15. Ela apresenta uma melhora de desempenho em precisão e revocação em comparação com modelos como *SVM*, *Decision Tree* e *Autoencoder*.

Adiban *et al.* [85] propõem um sistema de detecção de anomalias de rede que explora a capacidade de reconhecimento de padrões do modelo *GAN*. Eles enfatizam que sistemas não supervisionados são superiores, pois a dependência em dados rotulados não é prática e tem um alto custo. A solução introduz a arquitetura *STEP-GAN* com múltiplos geradores e uma nova função objetivo para resolver o problema inerente de *mode collapse*. O modelo é treinado com exemplos benignos e o discriminador é aplicado para separar tráfego normal do anômalo. O sistema é avaliado em duas bases desbalanceadas, ICS e UNSW-NB15, e mostra capacidade superior a trabalhos relacionados basedos em *OCAN*. Os autores ainda provam a generalização da solução conduzindo experimentos em outras sete bases de dados de detecção de anomalias.

O gerador e o discriminador podem ser combinados para implementar um processo híbrido de detecção de anomalias. Xu *et al.* [86] usam *GAN* baseada em transformador para detectar anomalias em séries temporais. As redes geradora e discriminadora são implementadas a partir de um mecanismo de transformador *encoder-decoder*, permitindo aprender correlações temporais dos dados de tráfego. O algoritmo de detecção calcula uma soma ponderada entre o erro de reconstrução e a saída do discriminador para o registro de tráfego em análise. Caso esse valor ultrapasse um certo limiar, o exemplo é rotulado como anômalo. Essa proposta apresentou uma melhora no *F1-score* em comparação com modelos clássicos como *PCA, RF, LSTM* e *MAD-GAN* para as bases SWT, WD e KDD99.

A Rede Adversária Generativa e suas variações têm evoluído progressivamente, tonando-se modelos complexos e de difícil entendimento. Komarchesqui *et al.* [67] propu-

seram um estudo que utiliza conceitos de *XAI* para agregar explicabilidade a um modelo *GAN-GRU* para detecção de anomalias de volume. Os autores executaram um algoritmo de seleção de atributos baseado em *SHAP* para determinar as variáveis de entrada mais importantes para detecção de *DDoS*. Os resultados experimentais mostram que somente um subconjunto dos atributos de entrada é necessário para detecção de ataques, provendo uma redução no custo computacional do modelo *GAN-GRU*.

A Tabela 1 apresenta um resumo sobre as principais características dos trabalhos revisados anteriormente. A última linha da tabela introduz o trabalho proposto nesta dissertação: um sistema de detecção de anomalias para redes baseadas em fluxo IP, como as *SDN*s. O *NIDS* utiliza o discriminador de uma rede adversária generativa tradicional para detectar anomalias de volume de tráfego [29]. Explora-se a aplicabilidade e desempenho do modelo de aprendizado profundo *1D-CNN* na implementação das redes discriminadora e geradora. Esse experimento permite estudar como a capacidade de aprendizado e geração de dados de uma *GAN* tradicional é afetada quando suas redes internas são construídas com camadas convolucionais. Trabalhos relacionados comumente requerem diversas variáveis de tráfego de entrada para detectar anomalias. Por outro lado, o sistema proposto utiliza apenas quatro variáveis para identificar anomalias, reduzindo o custo computacional do algoritmo de detecção. Os trabalhos correlatos da literatura comumente não abordam mitigação de intrusões, apresentando soluções que apenas detectam anomalias. O sistema proposto nessa dissertação possui um módulo responsável por mitigar tráfego anômalo de modo automático, preservando a qualidade dos serviços de rede sem intervenção humana. Os experimentos são conduzidos em três bases de dados públicas: Orion, CIC-DDoS2019 e CIC-IDS2017, e o sistema é comparado com modelos do estado da arte.

Tabela 1 – Comparação com os trabalhos relacionados.

Referência	**Ambiente**	**Base de dados**	**Entrada**	**Modelo**	**Mitigação**
[72] (2023)	*SDN-IoT*	CIC-DDoS2019	69	*LSTM*, *DNN*	✓
[73] (2023)	*SDN-IoT*	CIC-IDS2017	10	*LSTM*, *DNN*	✗
[74] (2021)	Tradicional	CIC-IDS2017, CSE-CIC-IDS2018	100	*LSTM*, *DNN*	✗
[56] (2023)	Tradicional	CICIDS2018, Edge_IIoT	77, 50	*DNN, CNN*, *BiLSTM*	✗
[75] (2022)	*IoT*	NSL-KDD, UNSW-NB15	122, 49	*DNN, SHAP*, *LIME, RuleFit*	✗
[76] (2023)	*SDN*	SDN Dataset	59	*MLP*, *DNN*	✗
[77] (2023)	*SDN*	InSDN, CIC-IDS2017	26, 42	*SVM, LR*	✗
[79] (2023)	*IoT*	NSL-KDD, CTU-IoT UNSW-NB15	41, 21, 43	*GAN, DNN*, *CNN, LSTM*	✗
[80] (2023)	Tradicional	NSL-KDD, BoT-IoT UNSW-NB15	40, 47	*AE, WGAN*, *XGBoost*	✗
[82] (2023)	*MQTT-IoT*	MQTT-IoT-IDS2020	N/A	*AE*, *GAN*	✗
[83] (2023)	*IoT*	UNSW-NB15, CIC-IDS2017	42, 78	*BiGAN*	✗
[84] (2022)	Tradicional	NSL-KDD, UNSW-NB15	40, 47	*AE, GAN*	✗
[85] (2023)	Tradicional	ICS, UNSW-NB15	128, 47	*GAN*	✗
[86] (2022)	Tradicional	SWaT, WADI, KDD99	51, 123, 34	Transformador, *GAN*	✗
[67] (2024)	*SDN*	CIC-DDoS2019	1, 6	*GAN-GRU*, *SHAP*	✗
Esta dissertação (2025)	*SDN*	Orion, CIC-DDoS2019, CIC-IDS2017	4	*1D-CNN*, *GAN*, *SHAP*	✓

4 SISTEMA PROPOSTO

Neste capítulo, descreve-se o sistema proposto para detecção de intrusão de maneira semi-supervisionada. A solução é exemplificada em um ambiente *SDN*, como ilustrado na Figura 6. O *NIDS* consiste em quatro módulos básicos: coleta de tráfego, pré-processamento, detecção de anomalias e mitigação.

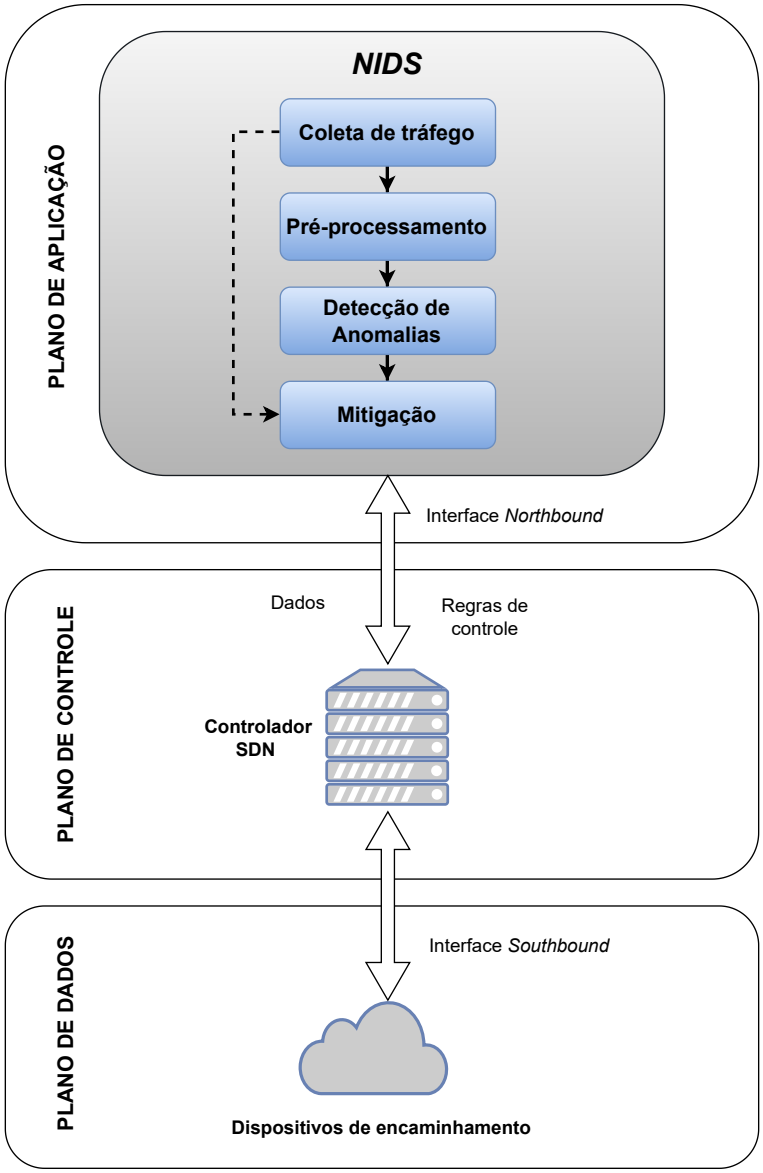

Figura 6 – Sistema de detecção de intrusão proposto para redes baseadas em fluxo IP.

4.1 Coleta de tráfego

Os dispositivos de encaminhamento do plano de dados armazenam estatísticas de tráfego. Essas informações podem ser coletadas pelo controlador *SDN* e acessadas através da interface *northbound*. O módulo de coleta de tráfego responsabiliza-se por requisitar dados de fluxo IP ao controlador para análise do sistema de detecção. A coleta de tráfego registrado na rede é realizada a cada segundo e encaminhada para o próximo módulo. O objetivo é reduzir o tempo de resposta do sistema a potenciais novas anomalias de rede.

4.2 Pré-processamento

Os registros coletados são pré-processados antes de serem analisados pelo *NIDS*. Os quatro passos de processamento necessários incluem tratamento de valores inválidos, cálculo de entropia, normalização de dados e atualização de janela deslizante.

4.2.1 Valores inválidos

As variáveis de fluxo IP podem possuir valores inválidos ou até mesmo valores nulos. Entre as razões para a corrupção de valores incluem-se erros de coleta e falha de hardware. Essas variáveis corrompidas não podem ser processadas pelo sistema e precisam ser corrigidas previamente. A cada coleta de tráfego, é necessário verificar e retificar possíveis erros nos dados. Portanto, as características numéricas dos fluxos IP coletados são analisadas em busca de valores vazios, infinitos ou *NaN* (*Not a Number*). Esses valores inválidos são substituídos pelo valor zero, com base nos resultados discutidos por Mall *et al.* [87].

A abordagem de imputação do valor zero tem o potencial de enviesar negativamente a base de dados por aumentar a quantidade de valores nulos de maneira não representativa. Apesar do uso desta estratégia nesta dissertação, o modelo 1D-CNN-GAN proposto supera os demais modelos comparados para as três bases de dados consideradas, alcançando eficácia em detecção e mitigação de intrusões, como discutido na seção 5. Estratégias mais representativas de imputação de valores, como as baseadas no cálculo de média e mediana das variáveis de fluxo IP, podem ser exploradas futuramente.

4.2.2 Entropia

O sistema proposto visa detectar anomalias de volume de tráfego, que podem ser identificadas analisando exclusivamente as distribuições de endereços IP e portas. Flutuações inesperadas nos níveis de troca de dados podem impactar a distribuição probabilística dos endereços IP e portas observados no tráfego. Como modelos de aprendizado profundo processam somente dados de entrada quantitativos, utilizou-se a Entropia de Shannon para resumir numericamente essas distribuições. O cálculo de entropia pode ser usado

para interpretar a dispersão e concentração de um conjunto de valores amostrados para uma certa variável qualitativa. A entropia do conjunto tende a aumentar quando os valores estão distribuídos uniformemente, uma vez que os dados estão dispersos. Por outro lado, quando poucos valores têm múltiplas ocorrências em comparação com os demais, a entropia do conjunto diminui devido à concentração dos dados.

Examinar mudanças nas entropias de variáveis qualitativas como endereços IP e portas é suficiente para identificar anomalias de volume, como discutido no capítulo 5. Diferentemente desta proposta, alguns trabalhos publicados pela comunidade científica apresentam soluções genéricas para o problema de detecção de intrusão [56], [76], [77]. Essas propostas necessitam de uma coleção mais ampla de variáveis de fluxo IP para detectar uma variedade maior de ataques, além dos de volume de tráfego. Esse aumento na dimensionalidade dos dados de entrada pode elevar o custo computacional do sistema.

A função (4.1) define a fórmula da Entropia de Shannon. x_θ representa o histograma de ocorrências dos possíveis valores que a variável qualitativa θ pode assumir, sendo $x_\theta = \{x_{\theta_1}, x_{\theta_2}, ..., x_{\theta_n}\}$. O termo x_{θ_i} indica o número de aparições do i-ésimo valor. O total de valores individuais observados para a variável θ é dado por S, definida como $\sum_{i=1}^{n} x_{\theta_i}$.

$$H(x_\theta) = -\sum_{i=1}^{n} \left(\frac{x_{\theta_i}}{S}\right) log_2 \left(\frac{x_{\theta_i}}{S}\right) \tag{4.1}$$

Os fluxos IP coletados pelo módulo anterior são convertidos para um registro de dados que resume o tráfego no segundo em análise. A conversão utiliza a função (4.1) para calcular os valores de entropia para o endereço IP de origem, endereço IP de destino, porta de origem e porta de destino. O módulo de detecção processa o registro composto por esses quatro valores para identificar anomalias no tráfego. Exemplificando, um ataque de negação de serviço distribuído visa sobrecarregar um serviço de rede com mais requisições que ele pode gerenciar, impossibilitando o acesso de usuários legítimos. É provável que, durante um ataque *DDoS*, o valor de entropia de endereço IP de destino seja abruptamente reduzido, já que o endereço da vítima é observado com maior frequência em comparação com os demais. Além disso, a entropia de porta de origem pode aumentar devido ao uso de portas aleatórias para condução do ataque, dispersando os valores observados. Ataques de escaneamento de portas (*Portscan*) vasculham as portas de um servidor alvo para identificar serviços disponíveis e encontrar vulnerabilidades. É possível que, durante a execução desse ataque, a entropia da porta de destino subitamente aumente devido ao grande número de portas distintas da vítima que recebem tráfego.

4.2.3 Normalização

A normalização de variáveis é um passo de pré-processamento comumente aplicado para treinar algoritmos de aprendizado de máquina de maneira eficiente [88]. O seu

propósito é padronizar os valores das diferentes variáveis de dados para que todas estejam contidas no mesmo intervalo numérico. Essa técnica facilita a convergência dos modelos e reduz o tempo de treinamento [56]. Como o sistema proposto é implementado utilizando redes neurais profundas, o registro de dados calculado previamente é normalizado. Diferentes problemas resolvidos por meio de aprendizado de máquina podem se beneficiar de métodos distintos de normalização. O método ideal para um dado problema é comumente encontrado através da aplicação de heurísticas ou experimentação.

Com base nos resultados discutidos por Radford *et al.* [89], normalizam-se as variáveis de entropia calculadas para o intervalo numérico [i, j], onde $i = -1$ e $j = 1$. A fórmula matemática para a aplicação da normalização está definida em (4.2) [90]. v' e v representam o valor normalizado e o original para uma dada instância de um certo atributo θ, respectivamente. min_θ indica o menor valor observado para θ, enquanto max_θ aponta o maior valor encontrado para esse atributo. O valor mínimo e máximo de θ são calculados com base no conjunto de dados de treinamento que contém somente entropias de tráfego legítimo. min_θ e max_θ são posteriormente reutilizados na fase de testes. Considerando a potencial concentração e dispersão de dados causada por eventos anômalos, entropias referentes a tráfego maligno podem possuir valores mínimos inferiores a min_θ e valores máximos que excedam max_θ. Nestes casos, o valor de entropia normalizado v' pode localizar-se fora do intervalo [-1, 1], auxiliando o módulo de detecção a identificar a anomalia.

$$v' = \frac{v - min_\theta}{max_\theta - min_\theta}(j - i) + i \tag{4.2}$$

4.2.4 Janela deslizante

A solução proposta detecta anomalias analisando as entropias de endereços IP e portas observadas em uma janela deslizante contendo os últimos q segundos. A abstração de janela deslizante é implementada utilizando uma estrutura de dados de fila de tamanho q. Essa estrutura é atualizada a cada segundo de análise, antes do estágio de detecção de anomalias. Para realizar a atualização, enfileira-se o novo registro de dados para o tempo atual t, removendo o registro mais antigo referente ao tempo $t - q$.

4.3 Detecção de anomalias

O módulo de detecção de anomalias é responsável por analisar a janela deslizante de tráfego e classificá-la como benigna ou anômala. São ilustrados na Figura 7 os seus três principais componentes. O *baseline* de rede representa o comportamento esperado do tráfego legítimo. O discriminador é uma rede neural que visa extrair o comportamento de um exemplo de janela deslizante que abrange o tempo $t - q + 1$ até o tempo atual

t. O limiar indica um limite de tolerância para a variação do comportamento observado. Primeiramente, esse módulo extrai os padrões comportamentais da janela usando a rede discriminadora, os quais são então comparados com o *baseline* de rede. Se a sua diferença exceder o desvio máximo de comportamento tolerado pelo sistema (limiar), o tráfego para o tempo atual de análise é inferido como anômalo. Nesse cenário, alertam-se os administradores de rede e o módulo de mitigação é acionado para conter a anomalia.

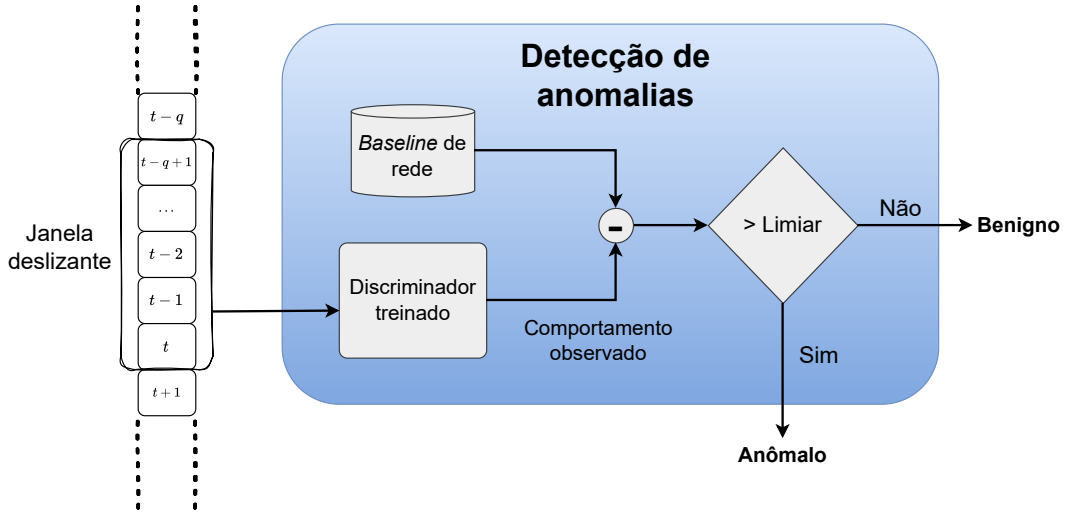

Figura 7 – Módulo de detecção de anomalias.

A configuração do módulo de detecção ocorre de maneira *offline* utilizando o Algoritmo 1. O objetivo é treinar uma Rede Adversária Generativa com dados históricos de tráfego benigno para aprender a sua distribuição estatística p_r. Após múltiplos experimentos, o valor máximo de iterações de treinamento foi definido como 20. Esse valor evita tempos excessivos de execução e permite que o modelo *GAN* alcance pontuações de validação aceitáveis. Durante 20 iterações, o conjunto de dados de janelas de tráfego benigno é embaralhado e dividido em lotes. Para cada lote, os pesos do discriminador e do gerador são atualizados uma vez, visando otimizar as suas respectivas funções objetivo, como descrito na função (2.1). O gerador busca minimizá-la, de modo a induzir o discriminador a interpretar exemplos gerados como sendo normais. O discriminador busca maximizá-la, pretendendo classificar suas entradas corretamente. Ao longo do treinamento, a rede geradora aproxima progressivamente a distribuição dos dados gerados p_g da distribuição dos dados reais. A rede discriminadora gradualmente se torna incapaz de diferenciar entre os exemplos gerados e os originais, mapeando instâncias de ambas as classes para uma probabilidade de saída próxima de 0,5. Generalizando, nessa etapa do aprendizado, o discriminador tende a mapear qualquer janela que siga os padrões da distribuição benigna p_r para um valor que se aproxima de 0,5. Por outro lado, a rede discriminadora potencialmente mapeia exemplos anômalos que não seguem p_r para um valor que se distancia de 0,5.

Algoritmo 1 Configuração *offline* do módulo de detecção de anomalias

Require: conjunto de treino e validação e rótulos de validação
Ensure: discriminador, *baseline* de rede e limiar

```
function VALIDAR()
    baseline ← média(discriminador(conjunto de treino))
    comportamentos ← discriminador(conjunto de validação)
    distâncias ← baseline - comportamentos
    limiar ← buscar valor entre 0,0 e 0,3
    predições ← "anômalo" para cada comportamento cuja distância seja maior que o limiar. Se não "benigno".
    mcc ← MCC(rótulos de validação, predições)
    return mcc

melhor_mcc ← 0
for época ← 1 até 20 do
    embaralhar conjunto de treino
    lotes ← dividir conjunto de treino em lotes de tamanho_lote elementos
    for lote em lotes do
        ajustar os pesos do discriminador.
        ajustar os pesos do gerador.
    mcc_validação ← VALIDAR()
    if mcc_validação > melhor_mcc then
        melhor_mcc ← mcc_validação
        melhor_discriminador ← discriminador
        melhor_baseline ← baseline
        melhor_limiar ← limiar
    if mcc_validação > 0.99 then
        Parar o treinamento.
```

Ao fim de cada iteração, executa-se um teste de validação para medir a convergência da *GAN* em relação ao aprendizado da distribuição de tráfego benigno. O teste aplica o modelo para configurar o módulo de detecção, como representado na Figura 7. O *baseline* de rede é dado pela média aritmética das saídas do discriminador para os exemplos de janelas de tráfego legítimo. Espera-se que esse valor esteja próximo de 0,5 caso a *GAN* atinja convergência. A rede discriminadora também é aplicada para extrair o comportamento de janelas amostradas do conjunto de validação, que contém tanto instâncias benignas quanto malignas. Por fim, busca-se um limiar que melhor separe as distâncias do *baseline* de comportamentos anômalos e benignos. O intervalo de busca é definido como $[0, 3 \times 10^{-1}]$, já que a rede discriminadora tende a produzir probabilidades de saída entre 0,5 e 0,8. O limiar escolhido é aquele que maximiza a métrica *Matthews Correlation Coefficient* (*MCC*) [91] para a classificação do conjunto de validação. Essa métrica foi escolhida por ser aplicável na avaliação da tarefa de classificação binária, além de apresentar resultados confiáveis em bases de dados desbalanceadas [92].

O treinamento do modelo se encerra quando a *GAN* alcançar uma pontuação de validação maior que 0,99 ou após a passagem de 20 iterações. O *baseline* de rede, o discriminador treinado e o limiar encontrado associado com o melhor valor de *MCC* de validação são salvos em disco para serem utilizados como configuração final do módulo de detecção.

Os experimentos mostram que o treinamento do modelo tradicional de *GAN* é de fato instável, como previamente apontado pela comunidade científica. Durante o treinamento, observou-se que os valores de *MCC* de validação tendem a oscilar bruscamente.

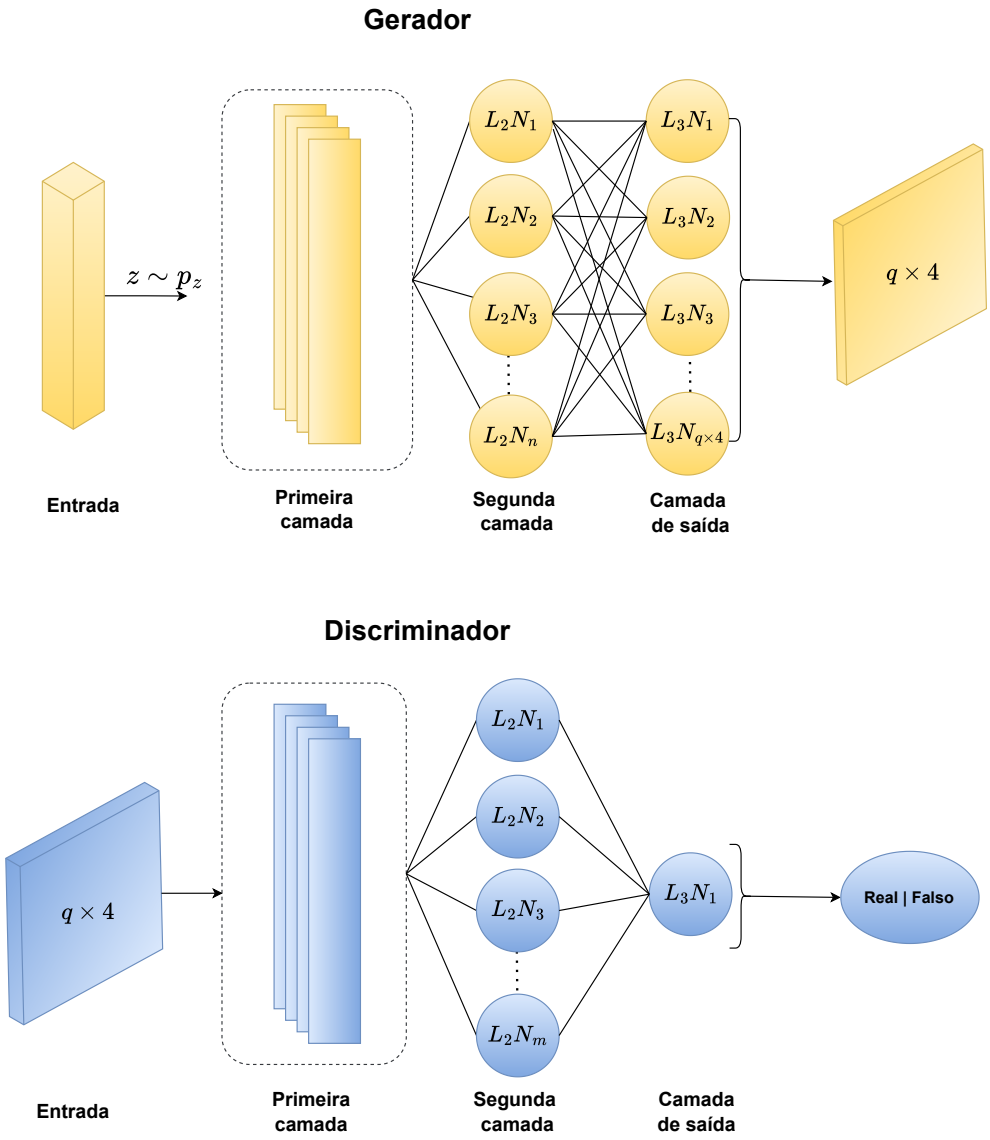

Figura 8 – Arquitetura do modelo *1D-CNN-GAN*.

4.3.1 Arquitetura do modelo *GAN*

Nessa dissertação, estuda-se a aplicabilidade e o desempenho do modelo de aprendizado profundo *1D-CNN* para implementar as redes discriminadora e geradora de uma *GAN*. Apresenta-se na Figura 8 a arquitetura proposta de Rede Adversária Generativa. O gerador recebe como entrada um vetor de ruído aleatório amostrado de uma distribuição p_z sobre um espaço latente de dimensionalidade d. Essa rede gera como saída uma janela deslizante de tráfego de tamanho q, que corresponde à entrada do discriminador. A rede discriminadora produz uma probabilidade que indica a chance da respectiva entrada ser

real ou sintética. Ambas as redes são compostas por duas camadas ocultas. A primeira utiliza convoluções unidimensionais para extrair padrões dos dados. A segunda camada é composta de neurônios tradicionais densamente conectados para calcular características de alto nível. Camadas de ativação *LeakyReLu* e *dropout* de 20% são incluídas após a primeira e segunda camadas ocultas para adicionar não linearidade e evitar *overfitting*, respectivamente. A taxa de regularização de *dropout* é definida como 20%, pois esse valor é comumente aplicado em trabalhos relacionados recentes, fornecendo resultados aceitáveis [84], [83]. Os pesos iniciais dos *kernels* unidimensionais da *1D-CNN* são definidos com base no inicializador *Glorot uniform* disponível na biblioteca de programação *Keras 2.11.0*. Durante a implementação, o modelo *1D-CNN-GAN* sofreu com instabilidade de treinamento. Após experimentos, o problema foi resolvido adicionando uma camada de normalização em lote após as camadas ocultas do gerador e discriminador.

4.3.2 Ajuste de hiperparâmetros

Os hiperparâmetros determinam a qualidade final de modelos de aprendizado e não podem ser encontrados mediante treinamento. Eles precisam ser cuidadosamente estudados e ajustados para o modelo final poder realizar a tarefa em questão. A estratégia de ajuste precisa ser executada utilizando um conjunto de dados de validação diferente do aplicado na fase de testes para evitar *overfitting* e promover a generalização do modelo. Existem duas abordagens básicas para o ajuste de hiperparâmetros: teste e erro e métodos de busca. Nessa dissertação, busca-se sistematizar o processo de configuração do sistema. Foi aplicado um algoritmo de busca exaustiva conhecido como *grid search* para ajustar os hiperparâmetros de treinamento da Rede Adversária Generativa de forma automática.

A Tabela 2 apresenta o espaço de hiperparâmetros explorado pelo algoritmo de busca. Por exemplo, para o tamanho de janela deslizante q, serão avaliados de maneira individual os valores inteiros 8 e 16 em combinação com os demais hiperparâmetros. Os valores das variáveis marcadas com um * foram definidos de acordo com heurísticas de treinamento discutidas por Radford *et al.* [89], com algumas exceções. Primeiro, a *LeakyReLu* como ativação das camadas ocultas promove uma variabilidade mais estável para a função objetivo da rede geradora do que a *ReLU* recomendada pelos autores. Segundo, o treinamento do discriminador progride mais rápido do que o do gerador ao aplicar uma taxa de aprendizado de $2,0 \times 10^{-4}$ para ambas as redes, levando à dissipação de gradientes. Esse problema é resolvido ao diminuir a taxa de aprendizado da rede discriminadora para $2,0 \times 10^{-5}$.

O total de combinações de valores a serem avaliados é dado pela multiplicação dos tamanhos dos espaços de busca individuais de cada hiperparâmetro. Portanto, o espaço de busca geral definido na Tabela 2 totaliza $2 \times 2 \times 2 \times 3 \times 3 \times 3 \times 3 \times 2 \times 2 = 2^5 \times 3^4 = 2592$ combinações de valores a serem testados. Durante a busca, cada combinação é usada para

Tabela 2 – Espaço de busca de hiperparâmetros.

Hiperparâmetro	**Espaço de busca**
Tamanho da janela deslizante (q)	8, 16
Dimensionalidade do espaço latente (d)	32, 64
Distribuição do espaço latente (p_z)	normal, uniforme
Unidades na primeira camada do gerador	8, 16, 32
Unidades na segunda camada do gerador	8, 16, 32
Unidades na primeira camada do discriminador	8, 16, 32
Unidades na segunda camada do discriminador	8, 16, 32
Tamanho de lote	128, 256
Otimizador	*Adam, RMSProp*
Método de normalização de dados*	min-max(-1,1)
Ativação da camada de saída do gerador*	tangente hiperbólica (*tanh*)
Ativação das camadas ocultas do gerador*	*LeakyReLu* com inclinação de 0,2
Ativação das camadas ocultas do discriminador*	*LeakyReLu* com inclinação de 0,2
Taxa de aprendizado do gerador*	$2,0 \times 10^{-4}$
Taxa de aprendizado do discriminador*	$2,0 \times 10^{-5}$

treinar a *GAN* e configurar o módulo de detecção com base no Algoritmo 1. Para cada modelo treinado, a pontuação de *MCC* para o conjunto de validação é registrada para inspeção futura. Ao final da busca, a combinação de valores de hiperparâmetros associada à maior pontuação de *MCC* é selecionada como configuração final do módulo de detecção. Discute-se na seção 5.2 os resultados do algoritmo de busca proposto.

4.4 Mitigação

O módulo de mitigação é responsável por gerenciar e atualizar duas listas de endereços IP: a lista segura e a lista de bloqueio. A primeira armazena os endereços IP de origem que recentemente produziram tráfego legítimo e não devem ser bloqueados. A segunda guarda os endereços IP de origem envolvidos em eventos anômalos recentes, que devem ser bloqueados visando impedir problemas de volume de tráfego. Esse módulo é acionado para todo segundo de análise e executa o Algoritmo 2. Ele recebe como entrada os fluxos IP coletados no último segundo, assim como a saída do módulo de detecção de anomalias. Se o tráfego atual for inferido como benigno, o módulo de mitigação adiciona na lista segura os endereços de origem presentes nesses fluxos. Se não, o algoritmo identifica os endereços de origem suspeitos, inserindo-os na lista de bloqueio. O módulo de mitigação também envia uma requisição ao controlador *SDN* requisitando o bloqueio dos endereços IP suspeitos. O controlador encarrega-se de definir as regras de descarte de tráfego potencialmente maligno, encaminhando-as para os dispositivos do plano de dados sem necessidade de intervenção humana.

Ambas as listas de gerência de endereços têm um parâmetro de tempo K que indica o número máximo de segundos que um certo endereço IP pode ser armazenado. O valor de K deve ser ajustado cuidadosamente, pois valores elevados podem ser prejudiciais caso o módulo de detecção faça inferências incorretas. Por exemplo, se o sistema de detecção

Algoritmo 2 Mitigação

Require: fluxos ← Fluxos IP coletados do último segundo de tráfego
Require: tráfego_é_benigno ← Saída do módulo de detecção
Require: lista_segura e lista_bloqueio
Ensure: atualização das listas

```
1: procedure ATUALIZAR_LISTA(endereços, lista)
2:     for endereço in endereços do
3:         if endereço não está na lista_segura nem na lista_bloqueio then
4:             K ← valor aleatório em [1, n]
5:             insere endereço na lista por K segundos

6: if tráfego_é_benigno then
7:     endereços_origem ← endereços de origem presentes nos fluxos
8:     ATUALIZAR_LISTA(endereços_origem, lista_segura)
9: else
10:    endereço_vítima ← endereço de destino mais frequente nos fluxos
11:    endereços_suspeitos ← endereços que se comunicam com endereço_vítima
12:    endereços_benignos ← endereços que não são suspeitos
13:    ATUALIZAR_LISTA(endereços_suspeitos, lista_bloqueio)
14:    ATUALIZAR_LISTA(endereços_benignos, lista_segura)
15:    requisitar_bloqueio(lista_bloqueio)
```

produzir um falso positivo, existe a probabilidade de que algum endereço IP benigno seja bloqueado por K segundos, caso ele ainda não esteja na lista segura. Falsos negativos também podem induzir o módulo de mitigação a adicionar algum endereço maligno à lista segura, permitindo que esse usuário gere tráfego anômalo livremente pelos próximos K segundos.

Outro cuidado que deve ser tomado ao definir esse parâmetro é a escolha de valores diferentes de K para endereços IP distintos. Quando um usuário legítimo começa a se comunicar na rede durante a ocorrência de um ataque, seu endereço IP ainda não está contido na lista segura. Caso ele envie tráfego para o endereço da vítima do ataque, é provável ser rotulado como suspeito e tenha seu endereço inserido na lista de bloqueio juntamente com os endereços malignos. Se K for igual para todos os registros, os endereços legítimos e malignos expiram da lista de bloqueio e reiniciam a sua comunicação na rede de maneira simultânea. Nesse cenário, o usuário benigno volta à situação inicial, sendo suscetível a ser bloqueado novamente. Para evitar esse problema, o módulo de mitigação seleciona um valor aleatório no intervalo $[1, n]$ para o parâmetro K associado a cada endereço IP em análise. Por meio de múltiplos experimentos, ajustou-se n para 10 segundos. A aleatoriedade da seleção reduz a probabilidade de diferentes endereços IP possuírem o mesmo valor de K. Assim, permite-se que um usuário legítimo bloqueado tenha a chance de voltar a comunicar-se na rede e seja adicionado à lista segura.

4.5 Complexidade Computacional

A análise de complexidade é uma área de estudo que visa entender o comportamento de algoritmos em cenários de pior caso. Os módulos de coleta e pré-processamento armazenam temporariamente e iteram em cada fluxo IP de entrada para executar as suas respectivas tarefas. Portanto, a sua complexidade temporal e espacial é $O(n)$, onde n indica o número de fluxos IP coletados para o instante de análise. Durante a operação do módulo de mitigação, o sistema mantém todos os fluxos coletados, processa cada um individualmente e acessa as listas segura e de bloqueio. Essas listas são implementadas utilizando uma estrutura de dados de dicionário, o qual provê tempo de acesso constante. Consequentemente, a complexidade de espaço e tempo do procedimento de mitigação também é $O(n)$. O consumo de recursos computacionais do módulo de detecção não cresce conforme o tamanho da entrada (fluxos IP) aumenta. Considerando que esse módulo processa uma janela deslizante de entrada de tamanho fixo, sua complexidade computacional é $O(1)$.

5 EXPERIMENTOS E RESULTADOS

Nesse capítulo, avalia-se a eficácia em detecção e mitigação de anomalias do sistema proposto. A solução é avaliada em bases de dados relevantes representando diferentes instâncias de redes baseadas em fluxos IP. De maneira geral, o sistema é projetado para operar em redes *TCP/IP* com suporte para exportação de fluxos IP.

As bases de dados publicadas pelo Instituto Canadense de Cibersegurança (*CIC*, do inglês *Canadian Institute for Cybersecurity*) e pelo grupo de pesquisa Orion foram utilizadas para configurar e testar o *NIDS* em ambientes de redes distintos. Segundo pesquisas recentes, os conjuntos de dados disponibilizados pelo *CIC* estão entre as fontes de dados mais utilizadas pela comunidade científica [42], [93], [94]. Elas são compostas de múltiplos tipos de ataques volumétricos e conjuntos amplos de características, fornecendo um cenário adequado para avaliar o sistema proposto nesta dissertação. Os experimentos são executados em um processador *Intel Core i7-4510U* com 8 GB de *RAM*. Os softwares utilizados incluem *Linux Mint 20, Python 3.10.9, Tensorflow 2.11.0, Keras 2.11.0* e *Scikit-learn 1.2.1*.

5.1 Bases de dados

5.1.1 Orion

O grupo de pesquisa Orion aplicou a ferramenta *Mininet* para emular uma Rede Definida por Software. Esse sistema é amplamente utilizado para criar *SDN* virtuais compostas de controladores, *switches, hosts* e as suas interconexões [95]. Os pesquisadores emularam uma rede de 6 *switches* organizados em uma topologia de árvore. Há um *switch* raiz, o qual está conectado a outros 5 *switches*. Esses nós intermediários conectam-se a 28 *hosts* cada, totalizando 140 máquinas na rede.

Eles aplicaram a ferramenta *Scapy* para sintetizar pacotes benignos de dados para serem enviados através da rede. O volume dos pacotes gerados varia ao longo do tempo para simular a evolução ciclo estacionária de tráfego, como comumente observado em redes reais [96]. A ferramenta *Hping3* é usada para injetar pacotes de *DDoS* na rede, enquanto a *Scapy* é aproveitada para criar ataques de escaneamento de portas.

A base de dados está disponível publicamente, sendo compostas por três dias completos de tráfego registrado no formato de fluxos IP [34]. O primeiro representa um dia de operação normal da rede. Já o segundo e o terceiro dia contêm instâncias de ataques *DDoS* e *Portscan* em diferentes intervalos. Em cenários reais, ataques são raros, representando somente uma pequena porção do tráfego total de rede. A base Orion é desbalanceada para

refletir a baixa frequência de ataques. Para fins de visualização, dividiu-se o primeiro e o terceiro dia em intervalos de um segundo e aplicaram-se os primeiros dois passos de pré-processamento apresentados na seção 4.2. São ilustradas na Figura 9 as variações de entropia de endereços e portas ao longo do primeiro dia. A Figura 10 apresenta as entropias para o terceiro dia. Essas variáveis também são visualizadas utilizando a técnica de redução de dimensionalidade *t-distributed stochastic neighbor embedding* (*t-SNE*). O *t-SNE* possibilita a visualização de registros multidimensionais em duas ou três dimensões. Essa técnica tem como foco a modelagem de padrões locais nos dados, preservando agrupamentos de dados (*clusters*) de uma mesma classe [97]. A Figura 11 mostra a visualização *t-SNE* para os registros de entropia do terceiro dia de tráfego. A formação de três *clusters* de dados para registros benignos, de *PortScan* e de *DDoS* ilustra a diferença comportamental entre essas três classes de dados.

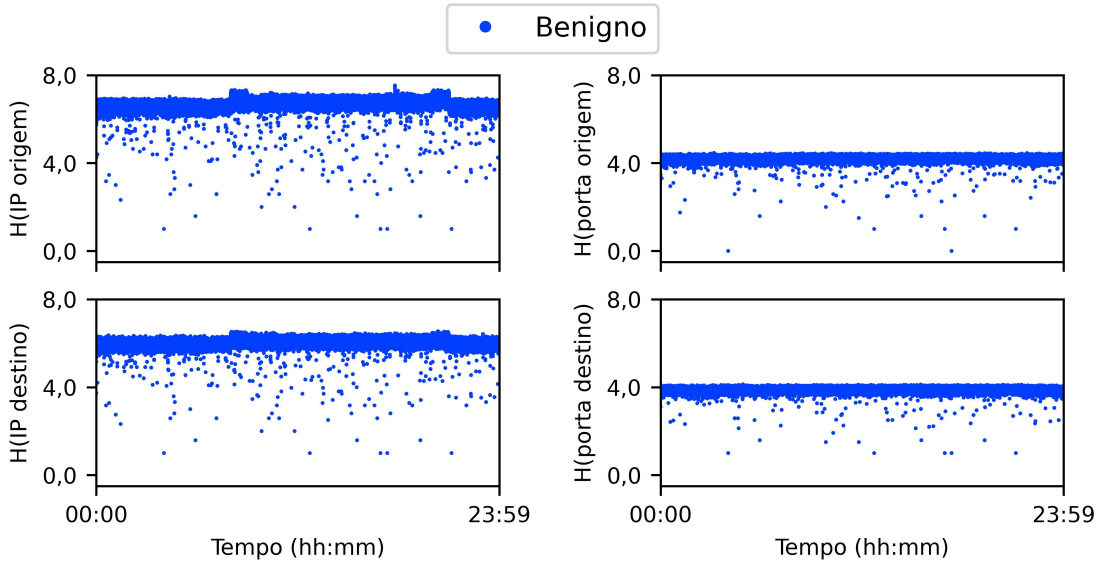

Figura 9 – Orion: entropias de endereços e portas para o primeiro dia.

5.1.2 CIC-DDoS2019

O Instituto Canadense de Cibersegurança criou e publicou a base de dados utilizada no segundo cenário de testes, a amplamente conhecida CIC-DDoS2019 [35]. Os cientistas implementaram uma rede vítima e uma atacante. A primeira é uma rede realista composta por um *firewall*, um roteador, dois *switches*, um servidor e quatro computadores pessoais. O tráfego benigno é gerado usando a abordagem *BProfile* proposta pelos próprios autores.

Essa base de dados contém dois dias de tráfego no formato de fluxo IP extraído de arquivos brutos de *pcap* a partir da ferramenta *CICFlowMeter*. A captura de tráfego referente ao primeiro dia inicia às 10:30 e finaliza às 17:15. Durante esse período, a rede

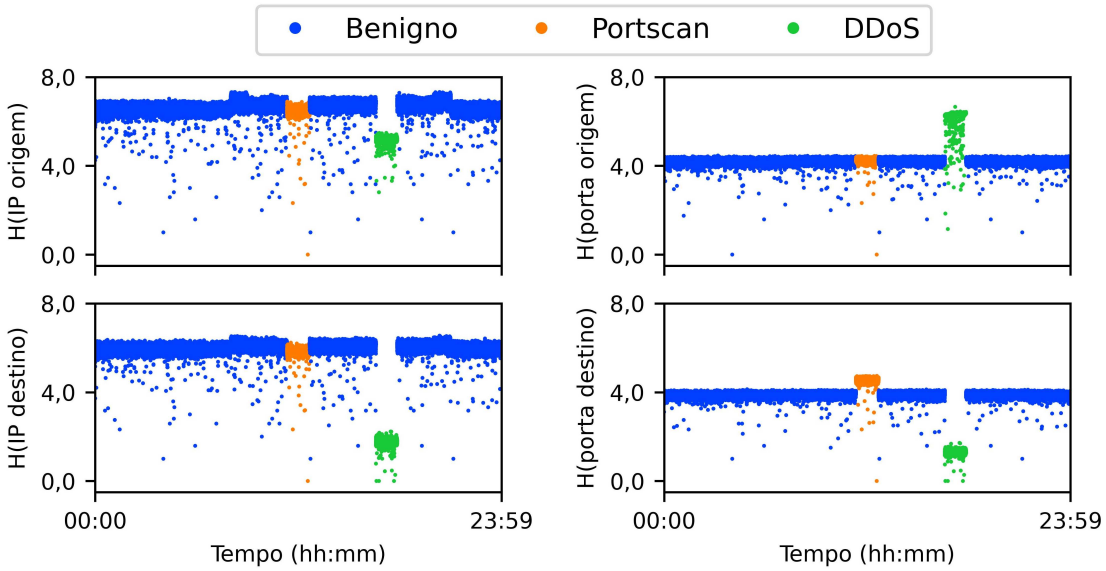

Figura 10 – Orion: entropias de endereços e portas para o terceiro dia.

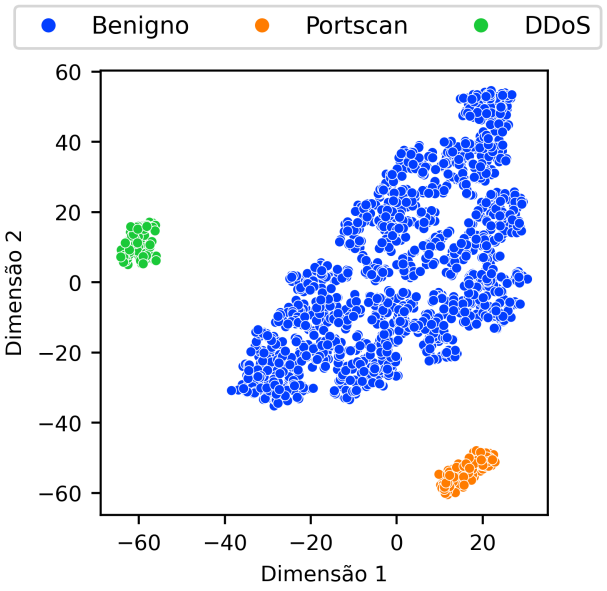

Figura 11 – Orion: visualização *t-SNE* para o terceiro dia.

vítima recebe tráfego de 12 diferentes tipos de ataques *DDoS* em diferentes intervalos: *NTP, DNS, LDAP, MSSQL, NetBIOS, SNMP, SSDP, UDP, UDP-Lag, WebDDoS, SYN* e *TFTP*. A coleta de tráfego do segundo dia inicia às 9:18, finalizando às 17:36. São lançados 7 tipos de ataques, incluindo *PortMap, NetBIOS, LDAP, MSSQL, UDP, UDP-Lag* e *SYN*.

Uma análise exploratória extensiva da base CIC-DDoS2019 evidencia a existência de diversos instantes de tráfego contendo quantidades mínimas de fluxos malignos. O

principal objetivo de um ataque *DDoS* é exaurir os recursos de uma rede ou servidor vítima, enviando mais tráfego do que pode ser processado pelo sistema. Durante um ataque *DDoS* fraco, os serviços alvos não são efetivamente negados e o comportamento normal da rede é meramente impactado. Exemplificando, no segundo dia de tráfego, o ataque *UDP-Lag* é executado durante 470s com uma frequência de aproximadamente 7 fluxos/s, totalizando 3120 fluxos e não afetando o funcionamento dos serviços de rede. Por outro lado, o ataque *Syn* é lançado na rede por 3791s, contando com 4 milhões e 898 mil fluxos. Cerca de 99% desse tráfego ocorre já nos primeiros 497s do ataque, alterando consideravelmente o volume de tráfego. O 1% restante de fluxos malignos são executados ao longo de 3294s com uma frequência de 4 fluxos/s, sem impactar o comportamento da rede. Apesar de não ser uma anomalia de negação de serviço, o ataque *Portmap* foi executado no segundo dia de tráfego durante 449s. Nos primeiros 426s, os fluxos malignos ocorrem com uma frequência aproximada de 5 fluxos/s. Já nos últimos 23s do ataque, essa taxa sobe para 7745 fluxos/s, apresentando alteração no padrão de normalidade das entropias de tráfego. Os ataques *UDP-Lag, Syn,* e *Portmap* foram desconsiderados durante os experimentos dessa dissertação. Eles apresentam uma frequência de envio de tráfego majoritariamente baixa, podendo enviesar a avaliação de um sistema de detecção de anomalias de volume.

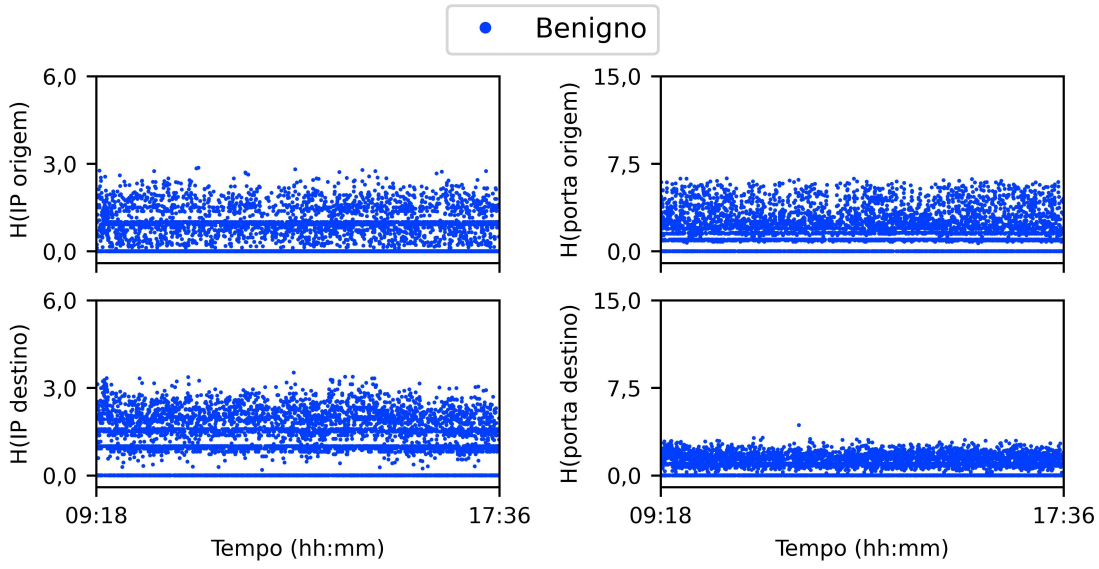

Figura 12 – CIC-DDoS2019: entropias de endereços e portas para o primeiro dia sem ataques.

Nesta dissertação, introduziram-se passos de processamento adicionais a essa base, além das mencionadas na seção 4.2. Primeiramente, ambos os dias contêm segundos com ausência de tráfego, os quais são efetivamente ignorados. Uma segunda versão do primeiro dia de tráfego é gerada, removendo os fluxos de ataque. O intuito é possibilitar o treinamento do modelo *GAN* com apenas padrões de comportamento legítimos. As entropias

para o primeiro dia sem ataques e o segundo dia são apresentadas na Figura 12 e na Figura 13. O gráfico *t-SNE* pode ser visualizado na Figura 14. As diferentes classes de ataque apresentam padrões numéricos distintos, agrupando-se entre si e contrastando com registros benignos.

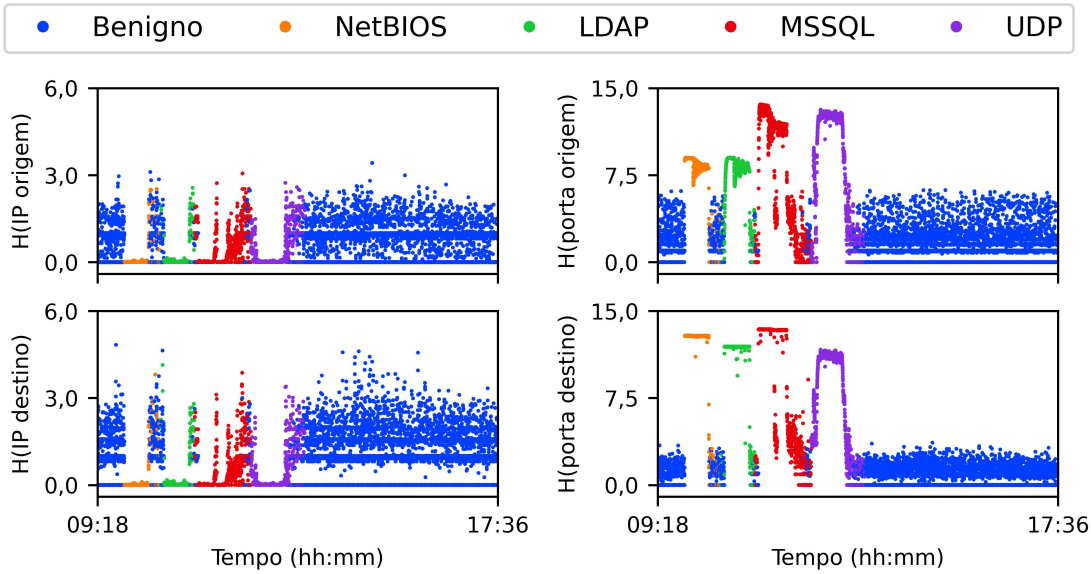

Figura 13 – CIC-DDoS2019: entropias de endereços e portas para o segundo dia.

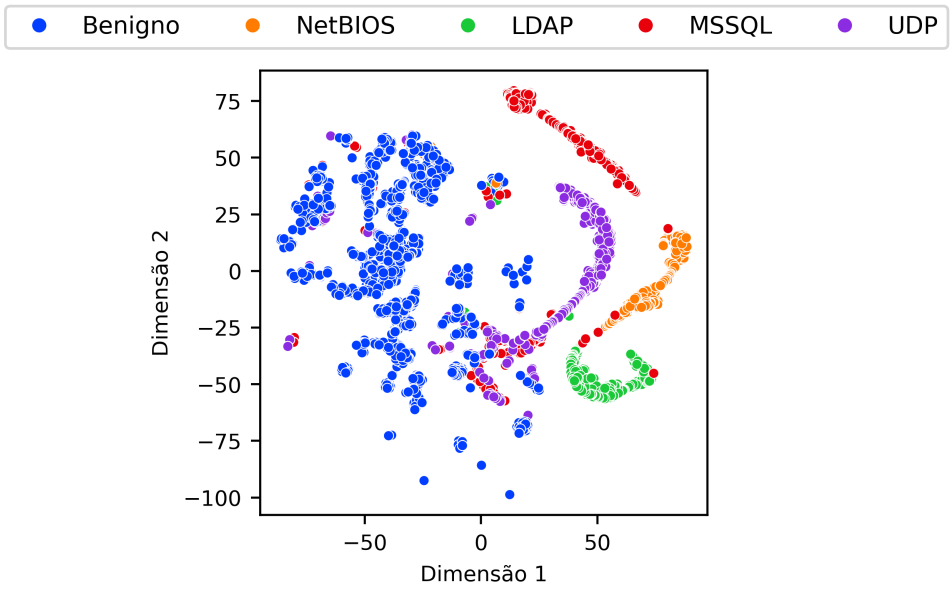

Figura 14 – CIC-DDoS2019: visualização *t-SNE* para o segundo dia.

5.1.3 CIC-IDS2017

O *CIC* também é responsável por publicar o terceiro cenário de redes considerado, o CIC-IDS2017 [36]. Essa base de dados contém uma semana de tráfego armazenado como fluxos IP. Foram selecionados três dias de tráfego para conduzir os experimentos propostos, dos quais dois contêm exemplos de ataques baseados em volume. A segunda-feira é livre de ataques, representando um dia de uso normal da rede. A quarta-feira contém algumas instâncias do ataque *DoS Hulk*. A sexta-feira possui ataques de *Portscan* e *DDoS LOIC* no período da tarde. Os registros de fluxo IP dessa base de dados possuem a indicação somente da hora e minuto de início da troca de dados. Essa característica impede que o sistema proposto por essa dissertação realize a sua análise a cada segundo, forçando-o a coletar o tráfego a cada minuto. As entropias para a segunda-feira e para a sexta-feira são ilustradas na Figura 15 e na Figura 16. Apresenta-se na Figura 17 o gráfico *t-SNE* para a sexta-feira. Devido à deturpação das variáveis de entropia durante eventos anômalos, os registros de *PortScan* e *DDoS* se distanciam dos registros normais.

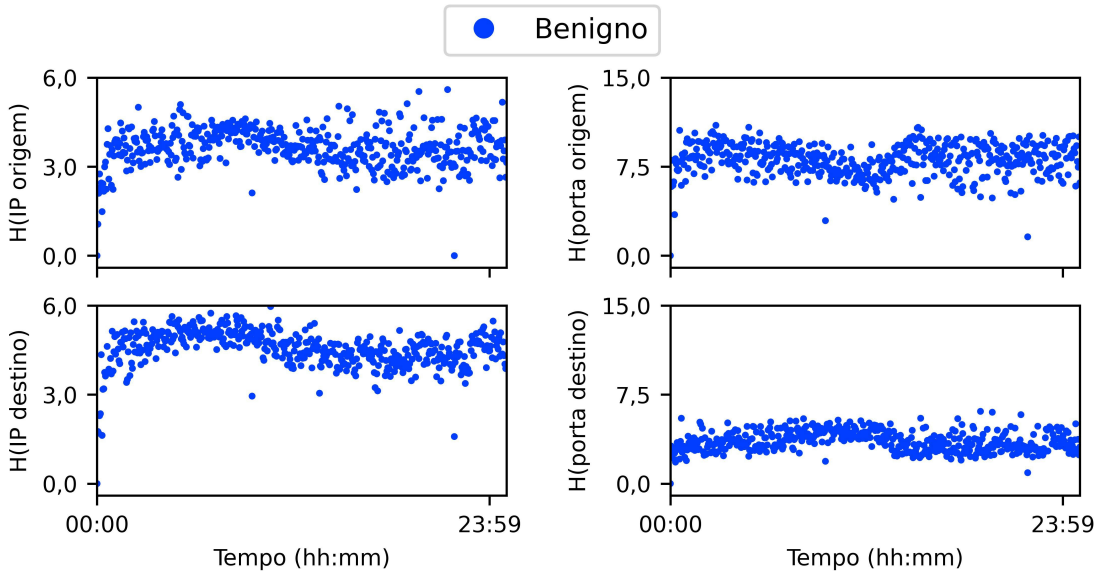

Figura 15 – CIC-IDS2017: entropias de endereços e portas para a segunda-feira.

5.2 Configuração do sistema

O sistema proposto é configurado nos três cenários de redes usando o algoritmo de busca de hiperparâmetros descrito na seção 4.3.2. Para a base Orion, o primeiro e segundo dia representam os conjuntos de treino e validação, respectivamente. Para a base CIC-DDoS2019, utilizou-se o primeiro dia sem ataques para treinar a *GAN* e o primeiro dia original para validar o modelo. A segunda-feira e a quarta-feira foram aplicadas, respectivamente, no treinamento e validação do modelo proposto para a base CIC-IDS2017.

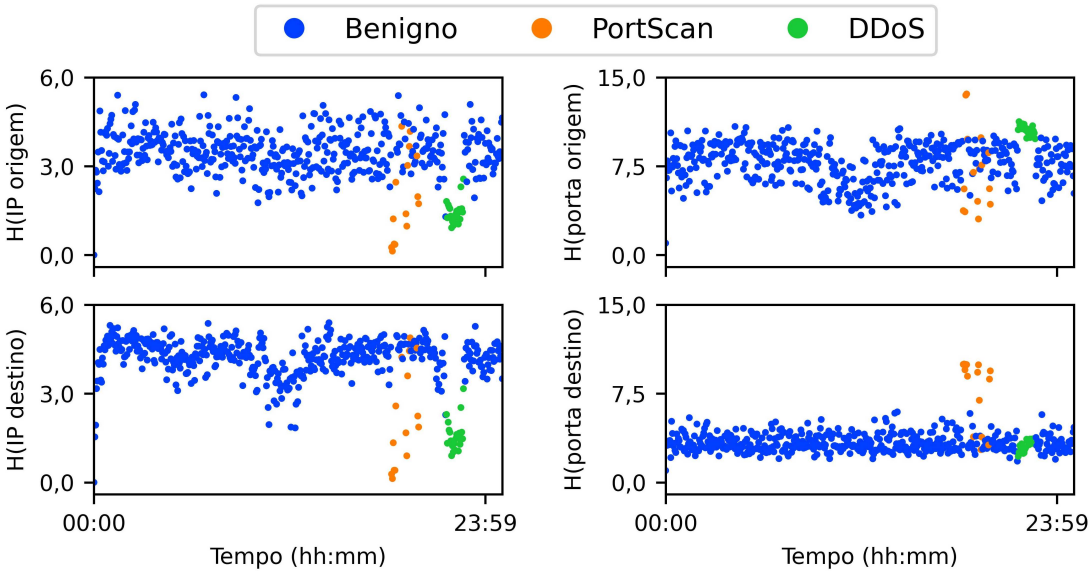

Figura 16 – CIC-IDS2017: entropias de endereços e portas para a sexta-feira.

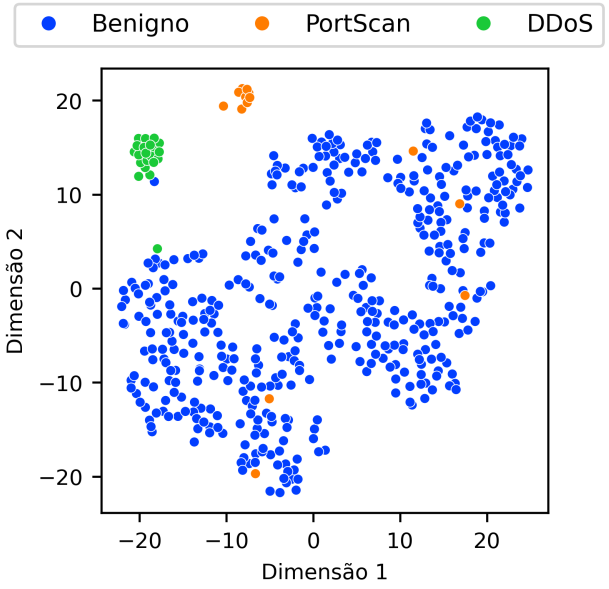

Figura 17 – CIC-IDS2017: visualização *t-SNE* para a sexta-feira.

A Tabela 3 apresenta os resultados da busca de hiperparâmetros, indicando a configuração otimizada para o modelo *1D-CNN-GAN* nas três bases. Os hiperparâmetros definidos mediante heurísticas foram omitidos dessa tabela, podendo ser consultados na Tabela 2. No cenário do CIC-IDS2017, foram avaliados tamanhos de janelas de 1 e 2, uma vez que cada elemento dessas estruturas já representa um minuto inteiro de tráfego de rede. Os tamanhos de lote nessa base foram redefinidos para 16 e 32, pois havia menos exemplos de dados disponíveis para treinamento em comparação com os cenários anteriores.

Foram implementadas redes adversárias generativas adicionais para fins de comparação com o modelo *1D-CNN-GAN*: a *LSTM-GAN* e a *TCN-GAN*. Essas redes são construídas de maneira semelhante à proposta dessa dissertação, diferenciando-se pelo modelo de aprendizado profundo utilizado na implementação do discriminador e do gerador. A *LSTM-GAN* utiliza uma *Long Short-term Memory* internamente, enquanto a *TCN-GAN* aplica uma *Temporal Convolutional Network*. A escolha desses modelos foi motivada por serem especializados no processamento de séries temporais, como dados de tráfego de rede. Na seção 5.2.2, será abordado como a capacidade de aprendizado e geração de dados da rede adversária generativa muda quando sua estrutura interna é implementada com esses diferentes modelos.

Tabela 3 – Hiperparâmetros ajustados para o modelo *1D-CNN-GAN*

Hiperparâmetro	**Orion**	**CIC-DDoS2019**	**CIC-IDS2017**
Tamanho da janela deslizante	16	8	2
Dimensionalidade do espaço latente	64	64	64
Distribuição do espaço latente	normal	normal	normal
Unidades na primeira camada do gerador	32	16	16
Unidades na segunda camada do gerador	32	16	16
Unidades na primeira camada do discriminador	8	16	16
Unidades na segunda camada do discriminador	8	16	16
Tamanho de lote	128	128	32
Otimizador	*Adam*	*Adam*	*Adam*

5.2.1 Análise de tempo de treinamento

Foram registrados os tempos de treinamento em segundos para os modelos baseados em *GAN*, como apresentado na Tabela 4. Esses tempos representam o pior caso, no qual os modelos são treinados até o final das 20 iterações, independente da pontuação de validação alcançada. Considerando a natureza estocástica desse processo, para cada combinação de modelo e base de dados, executou-se o algoritmo de treinamento três vezes e calculou-se o tempo médio de execução. O modelo *1D-CNN-GAN* é o mais rápido, sendo treinado em aproximadamente 58,36s, 24,43s e 15,13s nas bases Orion, CIC-DDoS2019 e CIC-IDS2017 (Tabela 4). A base Orion possui mais registros disponíveis para treinamento do que a base CIC-DDoS2019, já que, diferentemente da segunda, a primeira contém tráfego de rede durante todo o dia. Os resultados de tempo de treinamento dos modelos *LSTM* e *TCN* mostram que o segundo converge mais rápido que o primeiro quando o tamanho de entrada é pequeno (cenário CIC-DDoS2019). No entanto, quando o tamanho de entrada aumenta (cenário Orion), o tempo de treinamento do modelo *TCN* cresce, superando o do modelo *LSTM*. Portanto, o modelo com maior custo temporal no pior caso é o *TCN-GAN*. A base CIC-IDS2017 tem poucos registros de dados devido à organização de tráfego em intervalos de 1 minuto. Consequentemente, o tempo de treinamento é semelhante para os três modelos.

Tabela 4 – Tempo médio em segundos de treinamento no pior caso.

Modelo	**Orion**	**CIC-DDoS2019**	**CIC-IDS2017**
LSTM-GAN	137,60s	53,79s	17,68s
1D-CNN-GAN	**58,36s**	**24,43s**	**15,13s**
TCN-GAN	591,48s	46,55s	20,85s

5.2.2 Análise de *mode collapse*

Após o treinamento dos modelos generativos, foi analisada a proximidade visual entre a distribuição aprendida pelo gerador p_g e a distribuição do tráfego benigno p_r para o cenário CIC-DDoS2019. Para cada modelo, primeiro aplicou-se o gerador treinado para sintetizar janelas deslizantes de tráfego. Extraíram-se de cada exemplo de janela gerado os registros de entropia que a compõem, os quais representam p_g. Na sequência, foram amostrados registros aleatórios do conjunto de treinamento para representar p_r. Por fim, ilustrou-se p_g e p_r utilizando um plano cartesiano para cada par individual de entropia, permitindo a visualização da relação entre essas duas distribuições. As Figuras 18, 19 e 20 apresentam graficamente a relação entre diferentes combinações de entropias para os modelos *LSTM-GAN*, *1D-CNN-GAN* e *TCN-GAN*. Os valores de entropia estão normalizados para o intervalo [-1, 1], como discutido na seção 4.2.3.

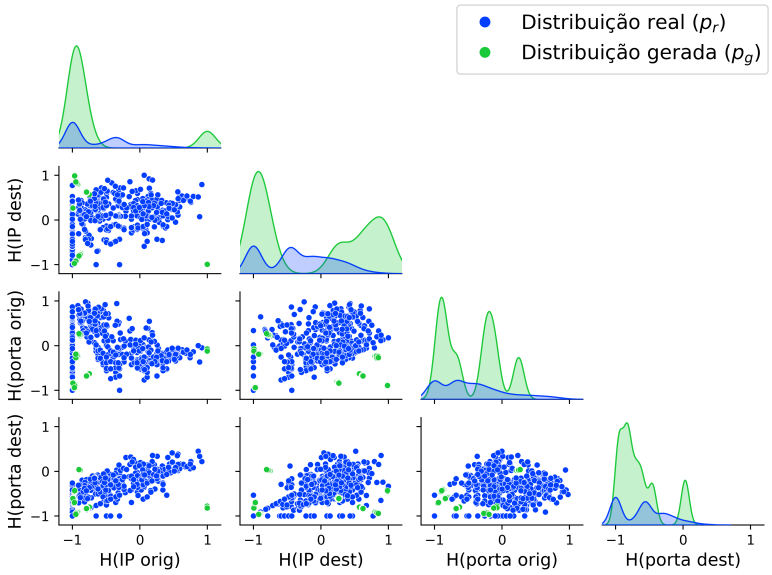

Figura 18 – Distribuição p_g aprendida pela *LSTM-GAN* para a base CIC-DDoS2019.

Os pontos azuis descrevem pares de entropias amostrados da distribuição de dados reais e os verdes refletem a distribuição gerada. O modelo *LSTM-GAN* sofre de *mode collapse* grave, ao aprender somente uma pequena parcela da distribuição p_r. O modelo *TCN-GAN* também não consegue capturar os padrões dos dados reais, mas o seu caso de *mode collapse* é ainda mais crítico: os pontos aprendidos quase não possuem interseção com os pontos reais. Por outro lado, a Figura 19 mostra que o modelo *1D-CNN-GAN*

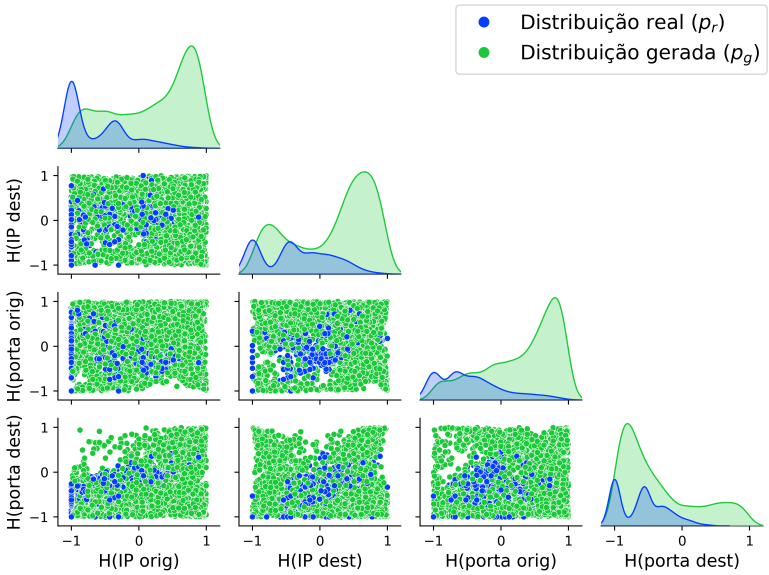

Figura 19 – Distribuição p_g aprendida pela *1D-CNN-GAN* para a base CIC-DDoS2019.

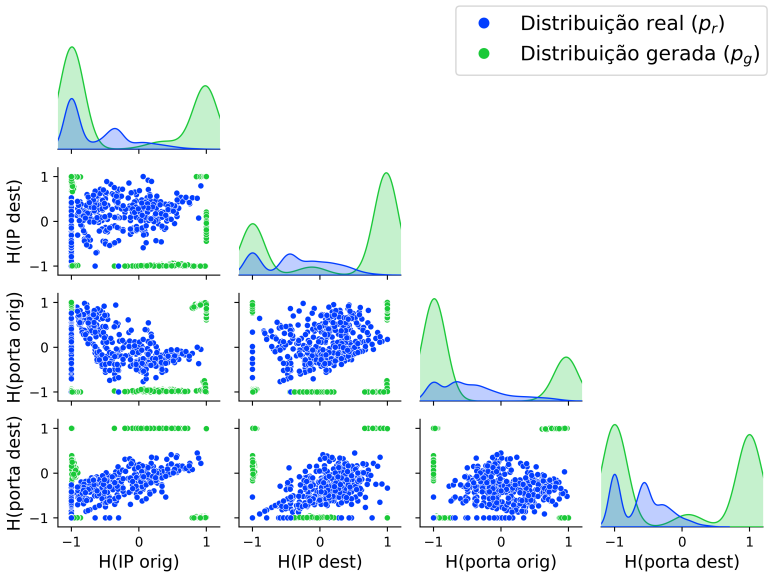

Figura 20 – Distribuição p_g aprendida pela *TCN-GAN* para a base CIC-DDoS2019.

aprende a maioria dos pontos dos dados reais e ainda os generaliza, incluindo pontos extras ao redor da distribuição dos reais. Conclui-se que os modelos *LSTM-GAN* e *TCN-GAN* são inadequados para geração de dados, enquanto o modelo baseado em convoluções pode ser útil nessa tarefa. Apesar de serem afetados pelo *mode collapse*, a capacidade discriminativa das redes baseadas em *LSTM* e *TCN* não é reduzida, como discutido na seção 5.3. Esses modelos alcançam pontuações de *F1-score* e *MCC* superiores a outros modelos de redes neurais implementados.

O desempenho de geração de dados do modelo *1D-CNN-GAN* pode ser explicado

pela sua similaridade com a *GAN* convolucional profunda (*DCGAN*, do inglês *Deep Convolutional Generative Adversarial Network*) [89]. O modelo *DCGAN* é conhecido pela sua capacidade de aprendizado de representação e prevenção de *mode collapse*. Assim como o modelo *DCGAN*, a implementação *1D-CNN-GAN* executa operações convolucionais em ambas as redes internas; possui normalização em lote para estabilidade de treinamento; usufrui da função de ativação *LeakyReLu*; evita o uso de camadas de *pooling*; e utiliza o otimizador Adam. Apesar das semelhanças, a *1D-CNN-GAN* ainda tem suas particularidades, como o uso de uma única camada convolucional unidimensional no gerador e discriminador, além do uso de camadas densas.

5.3 Detecção de anomalias

O módulo de detecção de anomalias proposto atua como um classificador binário por receber tráfego de rede como entrada e classificá-lo como benigno ou anômalo. Os desempenhos em detecção dos modelos implementados são medidos, portanto, utilizando métricas comuns de classificação binária. Entre elas têm-se Acurácia, Revocação, *F1-score, MCC*, curva *ROC* e área sob a curva *ROC* (*AUROC*).

Essas métricas são calculadas com base nas inferências verdadeiras positivas (VP), verdadeiras negativas (VN), falsas positivas (FP) e falsas negativas (FN) do sistema. Acurácia (Eq. 5.1) representa a porcentagem de inferências corretas. Precisão (Eq. 5.2) indica a fração das inferências positivas que estão corretas. Já a Revocação (Eq. 5.3) especifica a porção da classe positiva que foi corretamente classificada. A média harmônica entre a Precisão e a Revocação é dada pelo *F1-score* (Eq. 5.4). As métricas acima produzem resultados no intervalo entre 0 e 1, enquanto o *MCC* tem como saída valores entre -1 e 1. O *MCC* pode ser interpretado como um resumo do desempenho de classificação considerando todas as inferências do sistema. O valor 1 sugere que o classificador é perfeito e o sistema não possui inferências falsas positivas ou negativas. A pontuação em *MCC* tende a 0 para classificadores aleatórios, ao passo que um valor de -1 corresponde a um classificador que produz classificações invertidas para sua entrada.

$$Acurácia = \frac{VP + VN}{VP + FP + VN + FN} \tag{5.1}$$

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

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

$$F1 - score = 2 \times \frac{Precisão \ \times \ Revocação}{Precisão \ + \ Revocação} \tag{5.4}$$

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

Redes neurais do estado da arte, como *LSTM*, *1D-CNN* e *TCN* foram implementadas de maneira individual para conduzir um estudo comparativo de desempenho com os modelos generativos previamente introduzidos. Essas redes foram configuradas como regressores semi-supervisionados para analisar registros de dados de entropia. Com o intuito de promover uma comparação justa, a sua lógica de inferência e limiarização são semelhantes à do sistema proposto baseado em *GAN*. Os hiperparâmetros do modelo *1D-CNN-GAN* também foram aplicados a essas redes.

Para o cenário Orion, o dia três foi considerado conjunto de teste para realizar a avaliação de desempenho. A Tabela 5 mostra os respectivos resultados das métricas consideradas. De modo geral, o modelo *LSTM-GAN* é superior aos demais, apresentando as melhores pontuações em *F1-score* e *MCC*: 0,9860 e 0,9844. O modelo *1D-CNN* se mantém competitivo com o *LSTM* uma vez que os seus valores de *F1-score* e *MCC* ficam somente 0,0123 e 0,0138 pontos abaixo, respectivamente. A razão para esse resultado é a Precisão reduzida de 0,9630 do modelo convolucional, causada pelo aumento nas inferências falsas positivas. O modelo *TCN-GAN* alcança uma Revocação de 0,8899. Esse resultado sugere que ele teve dificuldades em identificar corretamente todos os exemplos anômalos analisados, culminando em uma taxa maior de falsos negativos. O modelo atinge uma pontuação considerável em *F1-score* e *MCC*, equiparando-se às redes neurais não generativas.

Tabela 5 – Métricas de classificação binária para o dia de teste do cenário Orion.

Modelo	**Acurácia**	**Precisão**	**Revocação**	***F1-score***	***MCC***
LSTM-GAN	**0,9970**	**0,9993**	0,9730	**0,9860**	**0,9844**
1D-CNN-GAN	0,9942	0,9630	**0,9847**	0,9737	0,9706
TCN-GAN	0,9874	0,9929	0,8899	0,9386	0,9333
LSTM	0,9866	0,9694	0,9041	0,9356	0,9288
1D-CNN	0,9858	0,9629	0,9026	0,9318	0,9244
TCN	0,9862	0,9587	0,9115	0,9345	0,9272

Para o cenário CIC-DDoS2019, utilizou-se o dia dois como conjunto de teste para o experimento de detecção. A Tabela 6 apresenta os resultados. A Revocação de todos os modelos é reduzida devido ao aumento das inferências falsas negativas. Essas previsões incorretas geralmente são provenientes dos instantes finais dos ataques, onde o tráfego anômalo é drasticamente reduzido e o nível de trocas de dados retorna ao seu estado normal. Naturalmente, um sistema de detecção de anomalias de volume interpreta esses ataques sutis como comportamento normal. Os resultados apontam que a *LSTM-GAN* alcança as melhores pontuações em *F1-score* e *MCC*, enquanto a *1D-CNN-GAN* continua próxima nas métricas. A *TCN-GAN* perde para os demais modelos generativos e redes neurais básicas, obtendo resultados inferiores em *F1-score* e *MCC*. Assim, os modelos

generativos baseados em *LSTM* e *1D-CNN* ainda são melhores opções para a detecção de anomalias utilizando entropias de endereço IP e porta, em comparação com as redes neurais implementadas.

Tabela 6 – Métricas de classificação binária para o dia de teste do cenário CIC-DDoS2019.

Modelo	**Acurácia**	**Precisão**	**Revocação**	***F1-score***	***MCC***
LSTM-GAN	**0,9345**	0,9709	**0,8633**	**0,9139**	**0,8653**
1D-CNN-GAN	0,9029	**0,9817**	0,7733	0,8651	0,8048
TCN-GAN	0,8506	0,9810	0,6419	0,7760	0,7052
LSTM	0,8712	0,9710	0,7011	0,8142	0,7419
1D-CNN	0,8660	0,9541	0,7007	0,8080	0,7287
TCN	0,8793	0,9764	0,7174	0,8271	0,7583

Para o cenário CIC-IDS2017, a sexta-feira foi escolhida como conjunto de teste para a realização dos experimentos. A Tabela 7 mostra os resultados de classificação. O modelo *LSTM-GAN* continua como a melhor implementação em termos de *F1-score* e *MCC*, e o modelo *1D-CNN-GAN* ainda obtém a segunda melhor pontuação. Todos os modelos avaliados apresentaram desempenho inferior em comparação com os cenários anteriores. O sistema proposto nessa dissertação é baseado no processamento de pequenos intervalos de tráfego de rede. Aplicá-lo em um contexto de intervalos maiores, como os de 1 minuto nessa base de dados, pode camuflar picos breves de volume anômalo em meio ao tráfego legítimo contínuo, reduzindo o desempenho de detecção.

Tabela 7 – Métricas de classificação binária para o dia de teste do cenário CIC-IDS2017.

Modelo	**Acurácia**	**Precisão**	**Revocação**	***F1-score***	***MCC***
LSTM-GAN	**0,9813**	**0,9090**	0,8333	**0,8695**	**0,8604**
1D-CNN-GAN	0,9771	0,8378	**0,8611**	0,8493	0,8370
TCN-GAN	0,9481	0,6486	0,6666	0,6575	0,6295
LSTM	0,9606	0,8695	0,5555	0,6779	0,6768
1D-CNN	0,9419	0,7857	0,3055	0,4399	0,4677
TCN	0,9585	0,9000	0,5000	0,6428	0,6531

A curva *ROC* é uma métrica que ilustra a relação entre a Revocação e a taxa de falsos positivos (*FPR*, do inglês *False Positive Rate*) de um classificador binário para diferentes limiares de separação das classes negativas e positivas. Quanto mais a curva se aproxima do canto superior esquerdo do gráfico, melhor a capacidade do modelo em separar classes negativas e positivas, facilitando a definição de um limiar. Nesse caso, o classificador consegue alcançar uma alta pontuação em Revocação enquanto mantém o seu *FPR* baixo. A *AUROC* é uma métrica comumente aplicada como uma forma de resumo da curva *ROC*. O seu valor de saída aumenta em função da capacidade de separação de classes do classificador. A Figura 21 mostra a curva *ROC* e a *AUROC* para os modelos implementados nos três cenários de teste. O modelo *LSTM-GAN* é o que melhor separa as classes das bases Orion e CIC-DDoS2019, como mostram as suas curvas próximas ao canto superior esquerdo do gráfico e os seus valores de *AUROC*. Em geral, todos os seis

modelos alcançam uma pontuação em *AUROC* acima de 0,80 e 0,90 para os cenários, indicando que eles conseguem atribuir distâncias consideráveis entre classes negativas e positivas.

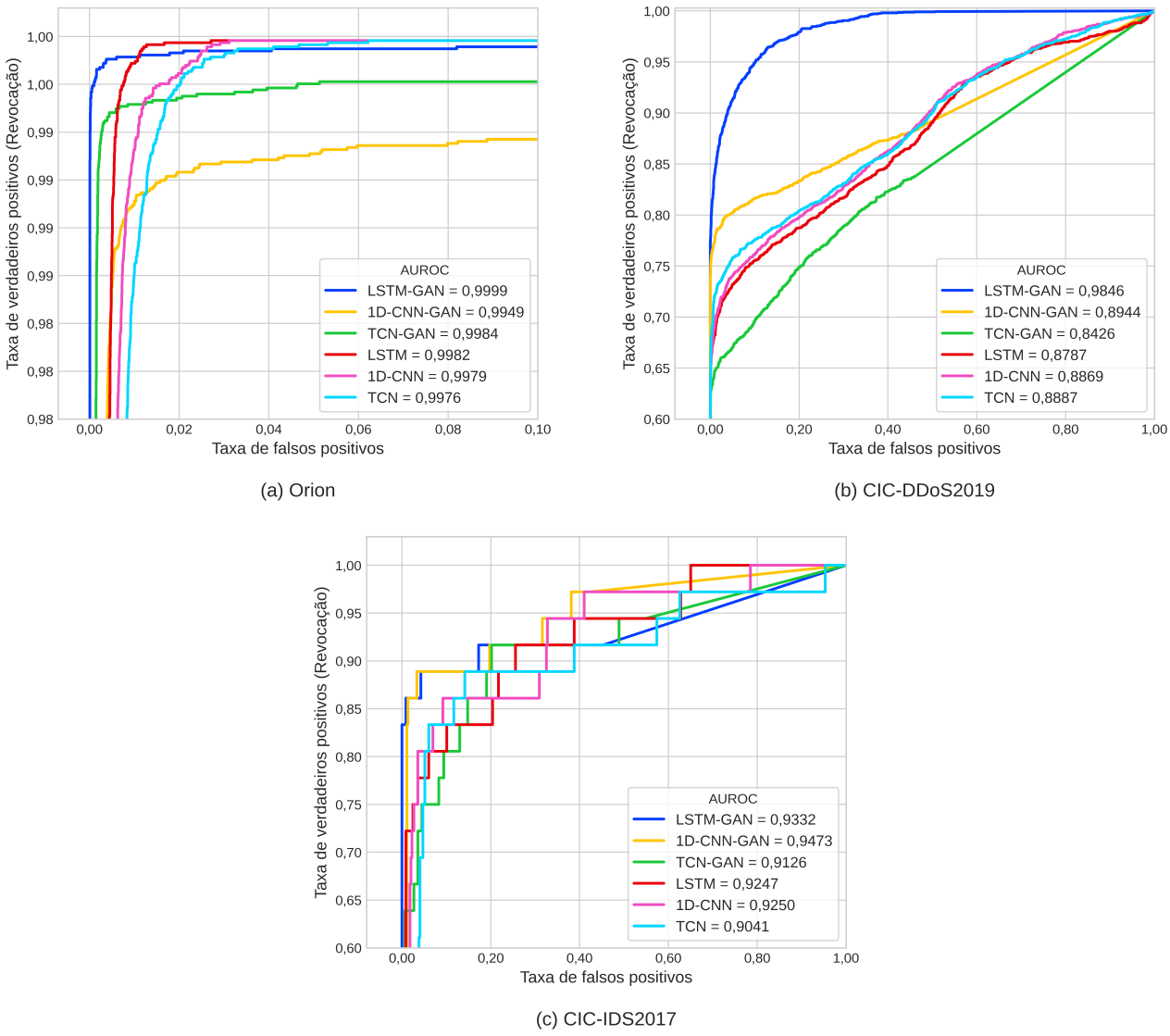

Figura 21 – Curvas *ROC*: Orion (a), CIC-DDoS2019 (b) e CIC-IDS2017 (c).

Destaca-se que o modelo *TCN-GAN* alcança um *MCC* de apenas 0,9333 no Orion, 0,7052 no CIC-DDoS2019 e 0,6295 no CIC-IDS2017 apesar de separar as classes de dados com um *AUROC* de 0,9984, 0,8426 e 0,9126 nessas mesmas bases de dados. Após investigação, descobriu-se que o limiar de detecção da implementação *TCN-GAN* sofre de *overfitting* no conjunto de validação, e, portanto, se torna menos representativo no conjunto de teste. A causa para esse problema encontra-se no fato de que os padrões de ataques presentes nesses conjuntos são diferentes. As redes neurais do estado da arte, *LSTM*, *1D-CNN* e *TCN*, também são vulneráveis a esse problema. Apesar da diferença de comportamentos anômalos no conjunto de validação, os limiares dos modelos *LSTM-GAN*

e *1D-CNN-GAN* não sofrem de *overfitting* e os seus desempenhos no conjunto de teste são relativamente mantidos. A implementação de um algoritmo de limiarização adaptável e não supervisionado se mantém como uma direção de pesquisa futura, visando evitar o potencial problema de *overfitting* e reduzir a taxa de inferências falsas positivas.

O modelo *LSTM-GAN* atinge o melhor desempenho geral no cenário CIC-DDoS2019 apesar de ter sofrido de um *mode collapse* severo, como ilustrado na Figura 18. Esse resultado sugere a seguinte hipótese a respeito do funcionamento interno dos modelos *GAN*s implementados: o *mode collapse* do gerador não impede o discriminador de aprender a distribuição dos dados reais p_r. Para checar a hipótese, alimentaram-se registros de cada ponto da distribuição p_r para o discriminador treinado da *LSTM-GAN*. A rede produziu uma saída próxima de 0,5 para todos os registros analisados, indicando que todos os exemplos seguem p_r e suportando a veracidade da hipótese. Mais experimentos devem ser conduzidos para checar esse fenômeno, sendo uma direção de pesquisa futura.

5.3.1 Análise de tempo de inferência

Calculou-se o tempo médio de inferência em segundos para os três modelos generativos, como apresentado na Tabela 8. Esses resultados se referem ao tempo necessário para classificar o conjunto de teste completo. As bases CIC-DDoS2019 e CIC-IDS2017 têm menos instâncias de dados do que a base Orion. Todos os modelos gastam entre 1 e 2 segundos para classificar os conjuntos de teste das bases *CIC* uma vez que a sua entrada é pequena. O seu comportamento para volumes de dados maiores (pior caso) é evidenciado no conjunto de teste da base Orion. O modelo *1D-CNN-GAN* ainda é o mais rápido, levando cerca de 3,11s para classificar os dados. Novamente, o modelo *LSTM-GAN* é mais rápido do que o *TCN-GAN*, por necessitar de aproximadamente metade do tempo para gerar as suas inferências.

Tabela 8 – Tempo médio em segundos de inferência para o conjunto de teste completo.

Modelo	**Orion**	**CIC-DDoS2019**	**CIC-IDS2017**
LSTM-GAN	10,00s	1,33s	**2,07s**
1D-CNN-GAN	**3,11s**	1,06s	2,08s
TCN-GAN	20,99s	**1,04s**	2,08s

Uma segunda análise de tempo médio de inferência foi conduzida. Esse teste visa investigar o custo temporal dos modelos generativos na classificação de uma única janela de entropia. Durante a execução do sistema proposto em tempo real, o tráfego de rede é coletado e analisado segundo a segundo. Portanto, para o sistema apresentar uma latência apropriada, o custo temporal total apresentado pelos módulos internos do *NIDS* deve ser inferior a um segundo para cada instante de análise. É imprescindível que o modelo de detecção aplicado possa processar um registro de dados de maneira eficiente e com o menor custo temporal possível. A Tabela 9 apresenta o tempo de classificação de registro

alcançado pelo modelo proposto e as demais redes generativas avaliadas. Verifica-se que a *1D-CNN-GAN* obtém o menor tempo nas três bases de dados, requerendo por volta de 5×10^{-3} segundos para processar uma janela de entropia. A *TCN-GAN*, novamente, apresenta o pior tempo dentre os modelos avaliados.

Tabela 9 – Tempo médio em segundos de inferência para uma única instância de dados.

Modelo	**Orion**	**CIC-DDoS2019**	**CIC-IDS2017**
LSTM-GAN	$1,85 \times 10^{-2}$s	$1,23 \times 10^{-2}$s	$8,10 \times 10^{-3}$s
1D-CNN-GAN	$\mathbf{5,30 \times 10^{-3}}$s	$\mathbf{5,50 \times 10^{-3}}$s	$\mathbf{5,20 \times 10^{-3}}$s
TCN-GAN	$1,75 \times 10^{-1}$s	$2,01 \times 10^{-1}$s	$2,11 \times 10^{-2}$s

A partir dos experimentos conduzidos, chegou-se à conclusão de que o *1D-CNN-GAN* é o melhor modelo por três motivos. Primeiro, ele apresenta uma solução promissora para o *mode collapse*, ao aprender uma distribuição p_g que se aproxima e generaliza p_r. Segundo, o modelo é o mais eficiente em termos de custo computacional para execução do treinamento e inferência. Terceiro, o seu desempenho em detecção de anomalias permanece próximo ao melhor resultado atingido pelo modelo *LSTM-GAN*. Portanto, os próximos experimentos serão direcionados exclusivamente ao modelo *1D-CNN-GAN*.

5.3.2 Análise *SHAP*

Aplicou-se o método *SHAP* de maneira global para entender como os segundos individuais de uma janela deslizante afetam a saída do discriminador do modelo *1D-CNN-GAN* para as bases Orion, CIC-DDoS2019 e CIC-IDS2017. Para executar esse experimento, selecionaram-se 200 exemplos de janelas do conjunto de teste. Na sequência, por simplicidade, extraíram-se os registros de entropia referentes somente aos segundos iniciais de cada janela. A partir desses registros, ilustrou-se o seu gráfico de resumo *SHAP*.

A Figura 22 mostra o gráfico de resumo *SHAP* para a rede discriminadora do modelo *1D-CNN-GAN* para o conjunto de teste Orion. Pode-se visualizar como a saída do discriminador é afetada pelas entropias de endereços IP e portas referentes aos quatro primeiros segundos das janelas selecionadas. Cada ponto no gráfico representa a relação entre uma instância específica de uma variável de entropia e a sua respectiva contribuição para a saída. O eixo x fornece a contribuição *SHAP*, que pode ser negativa, nula ou positiva. O eixo y indica a variável correspondente, onde H(θ)_t-i representa a entropia H para o atributo θ referente ao tempo t-i da janela deslizante. Os pontos estão pintados usando um mapa de cores que define se o valor da respectiva instância da variável é alto (azul) ou baixo (verde). O gráfico permite a visualização da distribuição de contribuições (valores *SHAP*) de cada variável para o conjunto de dados completo. Todas essas variáveis impactam consideravelmente a saída do modelo, pois seus valores *SHAP* estão espalhados pelo eixo x. O gráfico não fornece muita informação a respeito da diferença de influência de valores baixos e altos de entropia. Os pontos verdes e azuis estão, de modo geral,

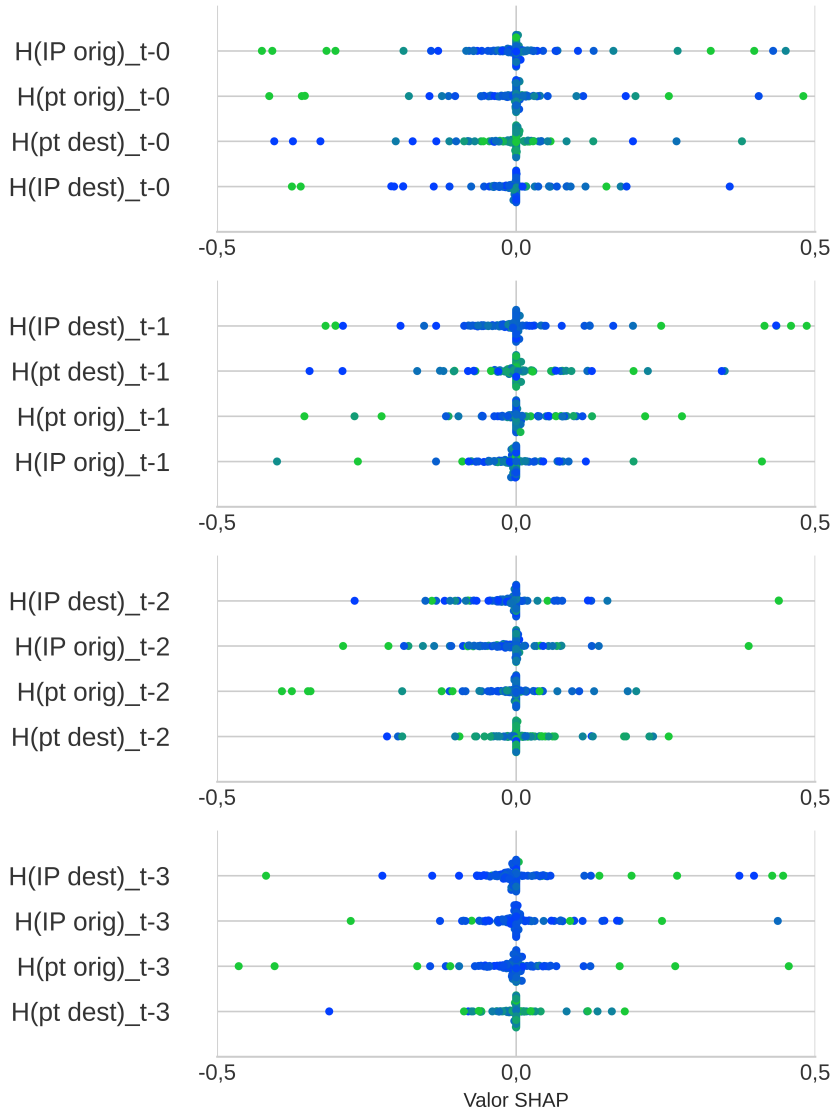

Figura 22 – Gráfico de resumo *SHAP* para o modelo *1D-CNN-GAN* no conjunto de teste Orion.

sobrepostos sem seguir algum padrão de distribuição. A aplicação de métodos adicionais de aprendizado de máquina explicável pode auxiliar na interpretação desse modelo para a base Orion.

A Figura 23 apresenta o gráfico de resumo *SHAP* para o discriminador da *1D-CNN-GAN* no conjunto de teste CIC-DDoS2019. Assim como para o cenário Orion, foram extraídos os registros de entropia referentes aos quatro primeiros segundos das janelas amostradas. Todas as variáveis de entropia impactam a saída da rede discriminadora. Os seus valores *SHAP* estão amplamente distribuídos no eixo x. Para os quatro segundos da janela, as variáveis de entropia de endereço IP tendem a ter um impacto maior (valor *SHAP* absoluto) na saída quando seus valores são baixos (verde). Como já apresentado na Figura 13, os ataques diminuem os valores de entropia de endereços, então é coerente que

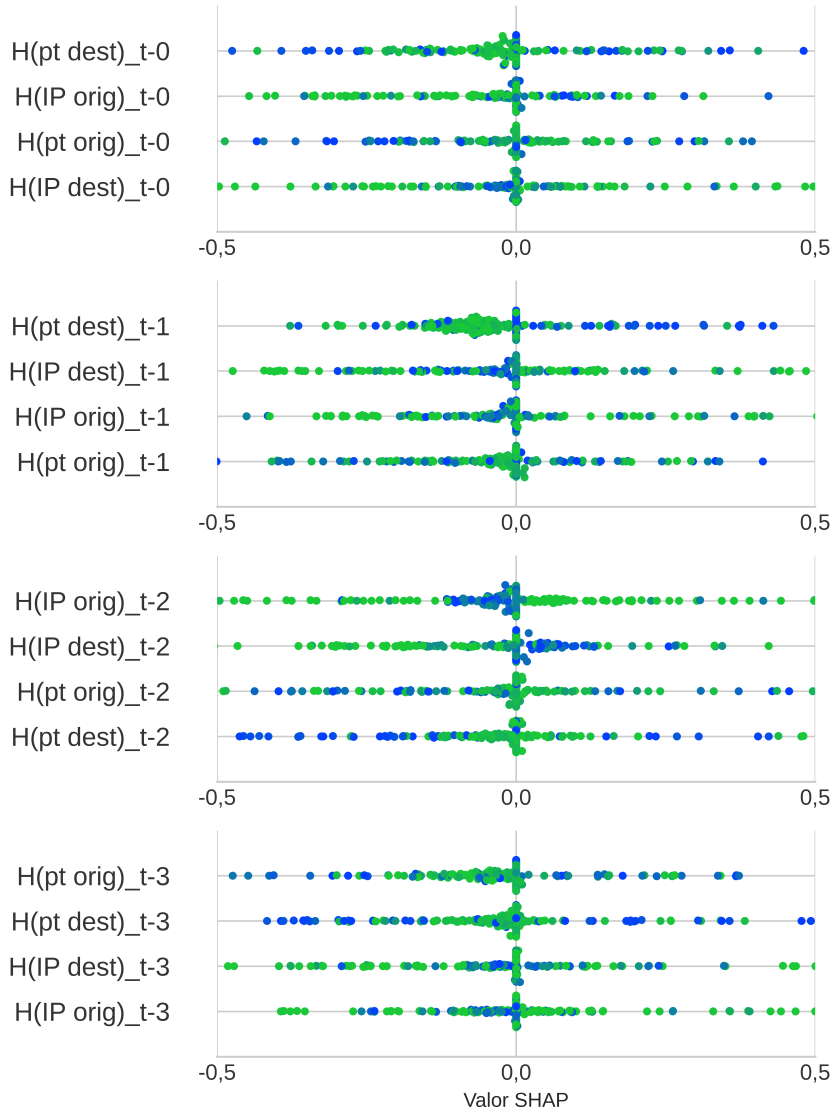

Figura 23 – Gráfico de resumo *SHAP* para o modelo *1D-CNN-GAN* no conjunto de teste CIC-DDoS2019.

o discriminador seja mais sensível a valores baixos nessas variáveis. Um comportamento similar é observado nas variáveis de entropia de portas, já que elas comumente têm mais influência na saída quando seus valores estão altos (azul). A Figura 13 indica que os ataques aumentam normalmente os valores de entropia de porta. Portanto, é compreensível que a rede discriminadora atribua mais peso a valores altos de entropia de porta.

A Figura 24 ilustra o gráfico de contribuições *SHAP* para a rede discriminadora do modelo proposto no conjunto de teste CIC-IDS2017. Foram extraídos os registros de entropia dos dois segundos que compõem cada janela amostrada neste cenário. A entropia de porta de destino é a única variável que apresenta comportamento (valor *SHAP*) consistente entre os tempos $t-0$ e $t-1$. Valores baixos (verdes) para essa variável impactam negativamente a saída produzida pelo discriminador, enquanto valores altos (azuis) têm

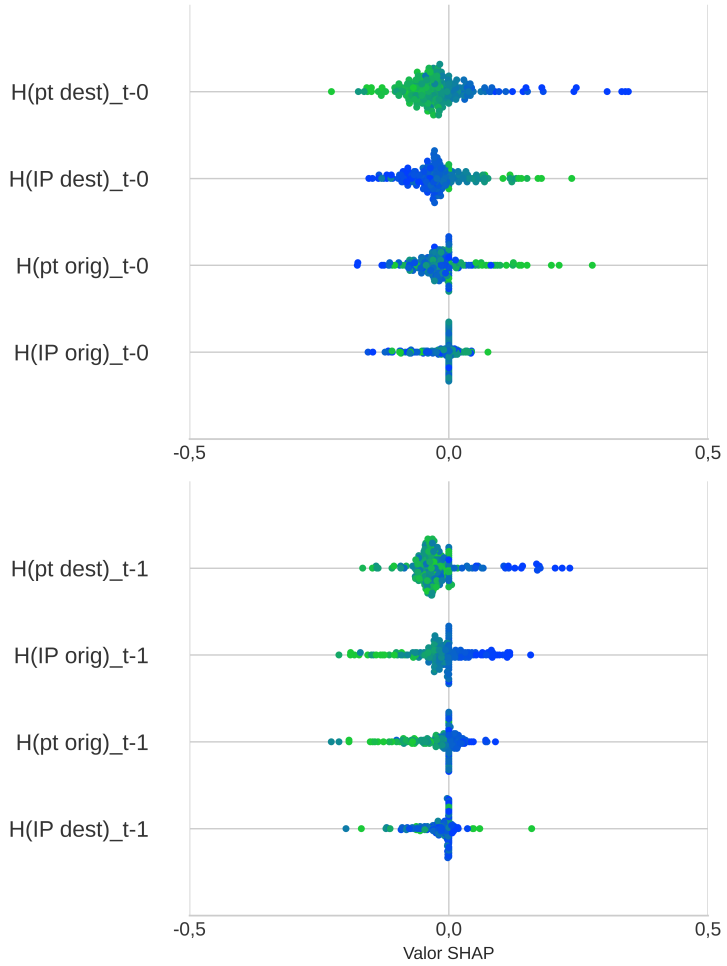

Figura 24 – Gráfico de resumo *SHAP* para o modelo *1D-CNN-GAN* no conjunto de teste CIC-IDS2017.

influência positiva. Segundo a Figura 16, instâncias de *Portscan* aumentam a entropia de porta de destino, explicando a influência positiva de valores altos para esta variável. As demais variáveis de entropias afetam a saída produzida pela rede discriminadora de maneira inconsistente. Por exemplo, valores baixos para entropia de porta de origem possuem impacto positivo no tempo $t-0$, porém exibem impacto negativo no tempo $t-1$. A falta de consistência entre os valores *SHAP* das entropias dificulta a realização de um estudo sobre a influência dessas variáveis na saída do discriminador.

A análise *SHAP* conduzida para as três bases de dados fornece pouco entendimento a respeito do impacto das janelas de entropia na saída do modelo *1D-CNN-GAN*. Para os cenários Orion e CIC-IDS2017, os valores *SHAP* mostram-se inconsistentes para os atributos avaliados. No cenário CIC-DDoS2019, os padrões de impacto são consistentes e refletem o comportamento dos ataques *DDoS* presentes no conjunto de testes. Os experimentos conduzidos sugerem que executar a análise *SHAP* no conjunto de dados completo pode gerar resultados inconclusivos. Portanto, como trabalho futuro, pretende-se aplicar

o gráfico de resumo *SHAP* a cada classe de tráfego de maneira separada para melhorar o entendimento do modelo. Essa análise visa avaliar de maneira isolada como registros de tráfego normais e anômalos impactam a saída da rede discriminadora. Outras técnicas de *XAI* também podem ser exploradas, como gráfico de dependência e mapa de calor [60], [61], visando alcançar a explicabilidade da rede discriminadora. Novos experimentos objetivando obter explicabilidade que considerem a ordem temporal das entropias na janela deslizante devem ser considerados trabalhos futuros. Esse tipo de estudo permite investigar anomalias que abrangem múltiplos segundos em termos do seu impacto no modelo 1D-CNN-GAN.

5.4 Mitigação

O desempenho de mitigação do sistema foi avaliado considerando a implementação *1D-CNN-GAN*. O módulo de mitigação foi executado para o conjunto de teste das bases de dados. A Tabela 10 mostra a quantidade de fluxos encaminhados e bloqueados para a base Orion. Mais de 99% de fluxos anômalos são bloqueados pelo sistema. A lista de bloqueio promove o correto descarte de fluxos maliciosos, apesar das inferências falsas negativas do módulo de detecção. Os 13.742 fluxos anormais não identificados representam uma fração mínima do tráfego de rede e são insuficientes para negar os serviços alvos, demonstrando a efetividade do sistema proposto. Os falsos positivos do módulo de detecção culminam no bloqueio de uma porção de 3% de tráfego benigno, apesar do uso do mecanismo de lista segura. A Figura 25 mostra a visualização dos registros de entropia para o conjunto de teste Orion após a aplicação do algoritmo de mitigação. Com a ação desse módulo, os padrões de tráfego permanecem próximos do comportamento de normalidade esperado, apesar da ocorrência dos ataques de *Portscan* e *DDoS*.

Tabela 10 – Resultado de mitigação na conjunto de teste Orion.

	Encaminhado	**Bloqueado**
Fluxos benignos	9.416.272 (96%)	302.030 (3%)
Fluxos anômalos	13.742 (<1%)	2.146.549 (99%)

A Tabela 11 apresenta o número de fluxos encaminhados e bloqueados para o conjunto de teste do CIC-DDoS2019. O sistema consegue bloquear cerca de 99% dos fluxos anômalos, encaminhando apenas 107.222 fluxos maliciosos. A lista de bloqueio ajuda a descartar endereços IP malignos, apesar das inferências incorretas do módulo de detecção. A lista segura auxilia o módulo de mitigação no tratamento das predições falso positivas, permitindo o encaminhamento de mais de 99% de tráfego benigno. Ilustra-se na Figura 26 a visualização dos registros de entropia para o conjunto de teste do CIC-DDoS2019 após a aplicação do processo de mitigação. Alguns fluxos maliciosos ainda poluem o tráfego benigno, porém eles não são suficientes para alterar o seu comportamento normal. Esse resultado comprova a eficiência dos módulos de detecção e mitigação.

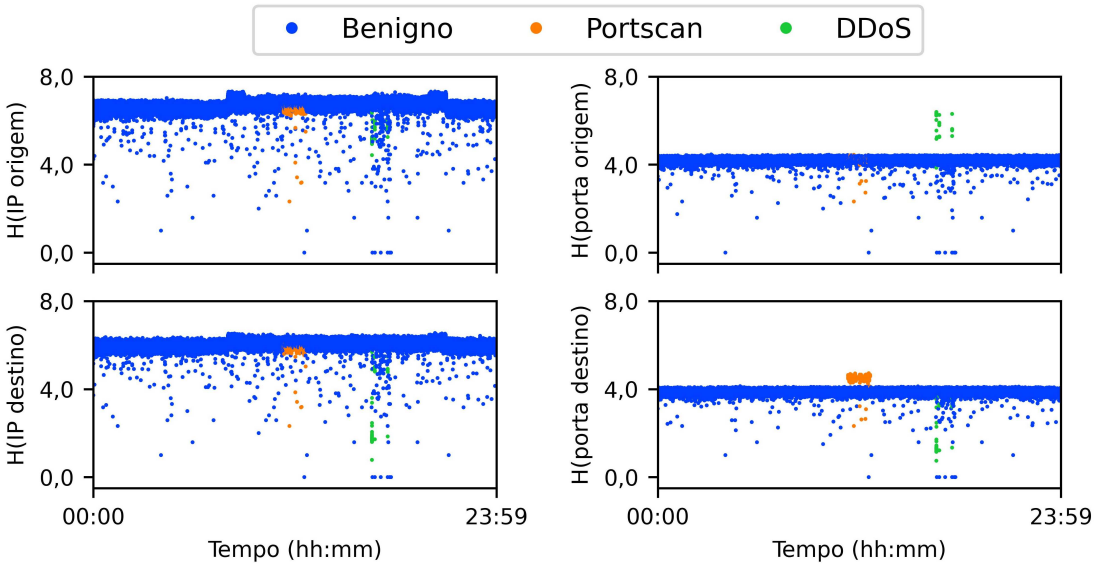

Figura 25 – Visualização dos registros de entropia após a mitigação para a base Orion.

Tabela 11 – Resultado de mitigação na conjunto de teste CIC-DDoS2019.

	Encaminhado	**Bloqueado**
Fluxos benignos	56.872 (99%)	93 (<1%)
Fluxos anômalos	107.222 (<1%)	15.120.005 (99%)

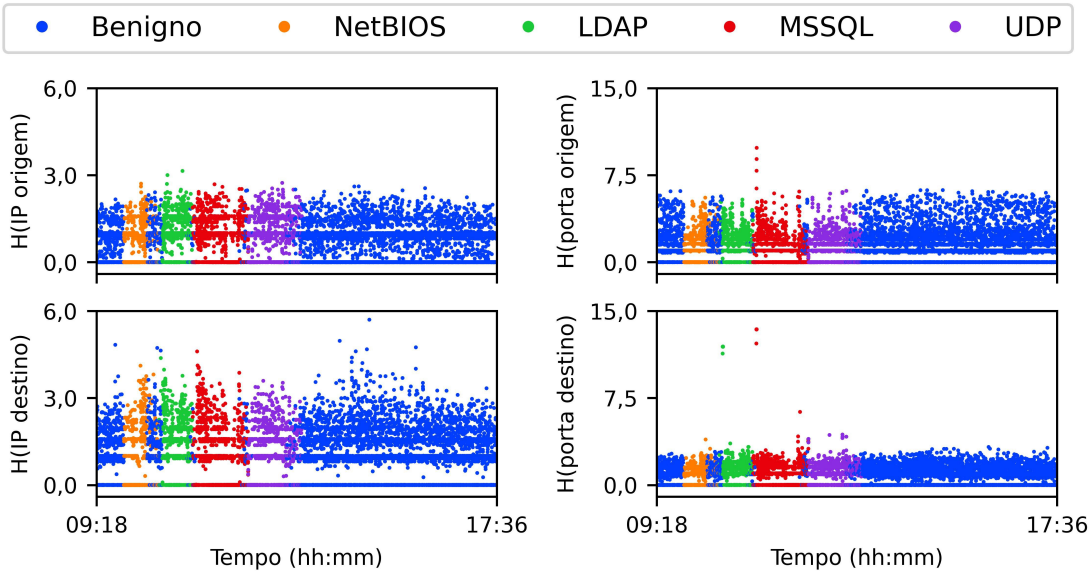

Figura 26 – Visualização dos registros de entropia após a mitigação para a base CIC-DDoS2019.

A Tabela 12 mostra o número de fluxos encaminhados e bloqueados para o conjunto de teste do CIC-IDS2017. Apesar de ser limitado pela análise de intervalos de 1 minuto, o sistema proposto consegue detectar e bloquear mais de 92% dos fluxos anôma-

los, encaminhando mais de 99% de tráfego legítimo. A Figura 27 apresenta a visualização dos registros de entropia para o conjunto de teste após a execução da mitigação. O ataque *DDoS* é contido, e os fluxos restantes encaminhados não têm impacto significativo no volume de tráfego. Os picos de ataque *Portscan* são quase todos descartados, com exceção dos fluxos de três minutos específicos.

Tabela 12 – Resultado de mitigação na conjunto de teste CIC-IDS2017.

	Encaminhado	**Bloqueado**
Fluxos benignos	415,218 (99%)	19 (<1%)
Fluxos anômalos	22,288 (7,8%)	264,297 (92,2%)

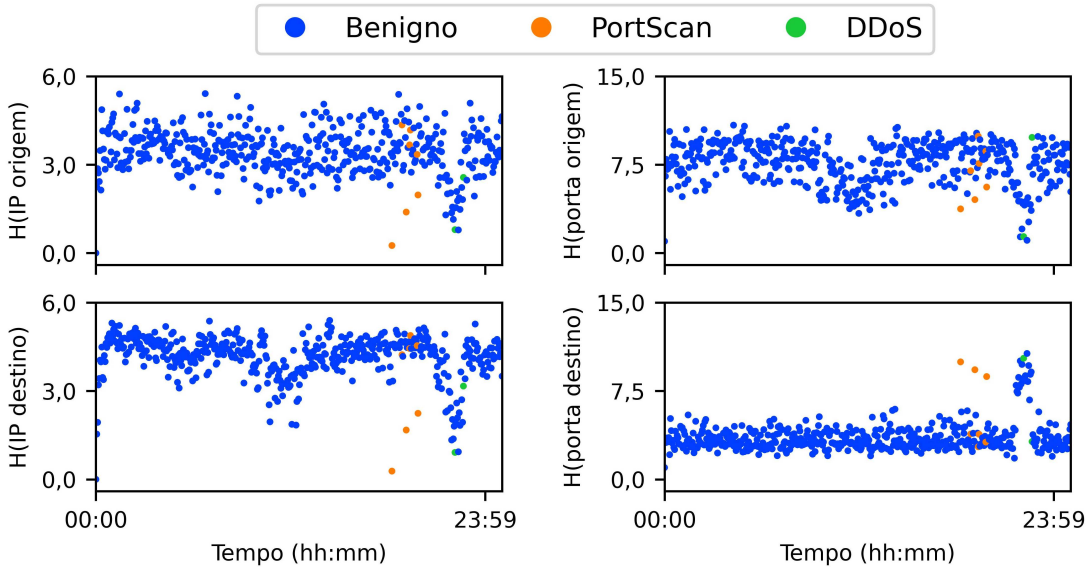

Figura 27 – Visualização dos registros de entropia após a mitigação para a base CIC-IDS2017.

6 CONCLUSÃO

Foi apresentado um sistema de detecção de intrusão semi-supervisionado para redes *TCP/IP* com suporte para coleta de fluxos IP. A solução identifica anomalias de volume de tráfego utilizando quatro variáveis de entropia de fluxo IP. Na configuração do *IDS*, treinou-se uma Rede Adversária Generativa tradicional baseada em convoluções (*1D-CNN-GAN*) para aprender o comportamento de tráfego legítimo, chamado de *baseline* de rede. O módulo de detecção de anomalias aplica o discriminador treinado para rotular todo padrão que desvie suficientemente do *baseline* como anômalo. O módulo de mitigação identifica e bloqueia automaticamente os endereços IP suspeitos envolvidos no evento anômalo. Os hiperparâmetros do sistema foram ajustados por meio de experimentação, heurísticas e algoritmo de busca.

Os experimentos para a avaliação do sistema foram executados em bases de dados públicas: Orion, CIC-DDoS2019 e CIC-IDS2017. Foram implementados os modelos alternativos *LSTM-GAN* e *TCN-GAN* visando avaliar e comparar o seu desempenho com o modelo proposto *1D-CNN-GAN*. Redes neurais como *LSTM*, *1D-CNN* e *TCN* também foram implementadas separadamente da rede generativa para fins de comparação.

O *1D-CNN-GAN* é o único modelo generativo capaz de resolver o problema de *mode collapse* e aprender uma p_g que se aproxima da distribuição dos dados reais p_r. Esse mesmo modelo é o mais eficiente em termos de custo computacional, enquanto o *TCN-GAN* é o pior por requerer tempos extensos de treinamento e inferência. Apesar de sofrer severamente de *mode collapse*, o modelo *LSTM-GAN* apresentou o melhor desempenho em detecção de anomalias em comparação com os outros modelos para os cenários de avaliação, atingindo *MCC*s acima de 0,86. O modelo *1D-CNN-GAN* alcançou um *MCC* mínimo de 0,80, sendo o segundo melhor em termos de desempenho de detecção. As curvas *ROC* calculadas para os modelos implementados nos cenários de teste revelam que eles conseguem separar adequadamente as classes benignas e malignas.

Os resultados sugerem que o modelo *1D-CNN-GAN* é o melhor considerando que ele resolve o *mode collapse*, é o mais eficiente computacionalmente, e alcança pontuações de detecção de anomalias competitivas com os demais modelos. Portanto, essa implementação é utilizada para avaliar o módulo de mitigação para os três cenários de teste. Os experimentos confirmam a efetividade do sistema proposto. A maioria dos fluxos anormais é bloqueada, enquanto uma pequena quantidade de tráfego legitimo é descartada. A lista de bloqueio e a lista segura auxiliam efetivamente o módulo de mitigação a lidar com as inferências falsas positivas e negativas do módulo de detecção.

Foi aplicada a técnica de inteligência artificial explicável *SHAP* para estudar o

impacto dos atributos de entropia na saída produzida pelo discriminador do modelo *1D-CNN-GAN*. Os experimentos foram executados no conjunto de teste das três bases de dados consideradas. De modo geral, os valores de influência *SHAP* apresentaram padrões numéricos inconsistentes nos cenários Orion e CIC-IDS2017. Essa inconsistência impossibilita o entendimento da relação entre a entrada e a saída da rede discriminadora. Por outro lado, no cenário CIC-DDoS2019, os valores *SHAP* são consistentes e permitem confirmar que a rede discriminadora é sensível a variáveis de entropia deturpadas por anomalias de volume. Portanto, verifica-se que a análise *SHAP* aplicada a conjuntos de dados compostos de múltiplas classes de registros pode gerar resultados inconclusivos. Executar esse experimento em conjuntos de registros contendo uma única classe de dados (normal ou anômalo) se apresenta como uma direção futura de pesquisa para melhorar a explicabilidade do modelo *1D-CNN-GAN*.

Como trabalho futuro, pretende-se desenvolver um método de limiarização dinâmico e não supervisionado para evitar o potencial problema de *overfitting*, como observado no modelo *TCN-GAN* e nas redes neurais avaliadas. Resolver esse problema poderia reduzir tanto a taxa de inferências falsas positivas como o bloqueio de tráfego legítimo, pois a eficácia em detecção influencia diretamente na robustez do módulo de mitigação. Novas bases de dados serão estudadas para avaliar o desempenho do sistema proposto em diferentes cenários de redes. Planeja-se aplicar técnicas e experimentos de *XAI* adicionais para melhorar o entendimento sobre o modelo proposto. Por fim, objetiva-se estudar o fenômeno de *concept drift* de tráfego de rede legítimo e como evitar a queda de desempenho do *NIDS* no longo prazo.

REFERÊNCIAS

[1] ERGEN, M. et al. Edge computing in future wireless networks: A comprehensive evaluation and vision for 6g and beyond. *ICT Express*, Elsevier, 2024. Disponível em: <https://doi.org/10.1016/j.icte.2024.08.007>.

[2] 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 BV, v. 192, p. 108076, jun. 2021. ISSN 1389-1286. Disponível em: <https://doi.org/10.1016/j.comnet.2021.108076>.

[3] IMRANA, Y. et al. A bidirectional lstm deep learning approach for intrusion detection. *Expert Systems with Applications*, Elsevier BV, v. 185, p. 115524, dez. 2021. ISSN 0957-4174. Disponível em: <https://doi.org/10.1016/j.eswa.2021.115524>.

[4] JAVED, Y. et al. Prism: A hierarchical intrusion detection architecture for large-scale cyber networks. *IEEE Transactions on Dependable and Secure Computing*, v. 20, n. 6, p. 5070–5086, 2023. Disponível em: <https://doi.org/10.1109/TDSC.2023.3240315>.

[5] SCARANTI, G. F. et al. Unsupervised online anomaly detection in software defined network environments. *Expert Systems with Applications*, Elsevier, v. 191, p. 116225, 2022. Disponível em: <https://doi.org/10.1016/j.eswa.2021.116225>.

[6] 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>.

[7] 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>.

[8] 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>.

[9] REGO, A. et al. Software defined network-based control system for an efficient traffic management for emergency situations in smart cities. *Future Generation Computer Systems*, Elsevier, v. 88, p. 243–253, 2018. Disponível em: <https://doi.org/10.1016/j.future.2018.05.054>.

[10] ASSIS, M. V. de et al. Holt-winters statistical forecasting and aco metaheuristic for traffic characterization. In: IEEE. *2013 IEEE International Conference on Communications (ICC)*. 2013. p. 2524–2528. Disponível em: <https://doi.org/10.1109/ICC.2013.6654913>.

[11] AYDıN, H.; ORMAN, Z.; AYDıN, M. A. A long short-term memory (lstm)-based distributed denial of service (ddos) detection and defense system design in public cloud network environment. *Computers & Security*, Elsevier BV, v. 118, p. 102725, jul. 2022. ISSN 0167-4048. Disponível em: <https://doi.org/10.1016/j.cose.2022.102725>.

[12] MUSTAPHA, A. et al. Detecting ddos attacks using adversarial neural network. *Computers & Security*, Elsevier BV, v. 127, p. 103117, abr. 2023. ISSN 0167-4048. Disponível em: <https://doi.org/10.1016/j.cose.2023.103117>.

[13] GUPTA, S. K.; TRIPATHI, M.; GROVER, J. Hybrid optimization and deep learning based intrusion detection system. *Computers and Electrical Engineering*, Elsevier BV, v. 100, p. 107876, maio 2022. ISSN 0045-7906. Disponível em: <https://doi.org/10.1016/j.compeleceng.2022.107876>.

[14] 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>.

[15] FRIHA, O. et al. Felids: Federated learning-based intrusion detection system for agricultural internet of things. *Journal of Parallel and Distributed Computing*, Elsevier BV, v. 165, p. 17–31, jul. 2022. ISSN 0743-7315. Disponível em: <https://doi.org/10.1016/j.jpdc.2022.03.003>.

[16] HIDALGO, C. et al. Detection, control and mitigation system for secure vehicular communication. *Vehicular Communications*, Elsevier BV, v. 34, p. 100425, abr. 2022. ISSN 2214-2096. Disponível em: <https://doi.org/10.1016/j.vehcom.2021.100425>.

[17] 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>.

[18] 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>.

[19] 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>.

[20] LOPEZ-MARTIN, M. et al. Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in iot. *Sensors*, MDPI, v. 17, n. 9, p. 1967, 2017. Disponível em: <https://doi.org/10.3390/s17091967>.

[21] PROENÇA, M. L. et al. The hurst parameter for digital signature of network segment. In: SPRINGER. *Telecommunications and Networking-ICT 2004: 11th International Conference on Telecommunications, Fortaleza, Brazil, August 1-6, 2004. Proceedings 11*. 2004. p. 772–781. Disponível em: <https://doi.org/10.1007/978-3-540-27824-5_103>.

[22] 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>.

[23] ARAFAH, M. et al. Anomaly-based network intrusion detection using denoising autoencoder and wasserstein gan synthetic attacks. *Applied Soft Computing*, Elsevier, v. 168, p. 112455, 2025. Disponível em: <https://doi.org/10.1016/j.asoc.2024.112455>.

[24] ITURBE-ARAYA, J. I.; RIFÀ-POUS, H. Enhancing unsupervised anomaly-based cyberattacks detection in smart homes through hyperparameter optimization. *International Journal of Information Security*, Springer, v. 24, n. 1, p. 45, 2025. Disponível em: <https://doi.org/10.1007/s10207-024-00961-6>.

[25] 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>.

[26] 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>.

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

[28] 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>.

[29] GOODFELLOW, I. et al. Generative adversarial nets. *Advances in neural information processing systems*, v. 27, 2014. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf>.

[30] 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>.

[31] CHAKRABORTY, T. et al. Ten years of generative adversarial nets (gans): a survey of the state-of-the-art. *Machine Learning: Science and Technology*, IOP Publishing, v. 5, n. 1, p. 011001, 2024. Disponível em: <https://doi.org/10.1088/2632-2153/ad1f77>.

[32] WANG, Y. et al. The application of evolutionary computation in generative adversarial networks (gans): a systematic literature survey. *Artificial Intelligence Review*, Springer, v. 57, n. 7, p. 182, 2024. Disponível em: <https://doi.org/10.1007/s10462-024-10818-y>.

[33] DUBEY, S. R.; SINGH, S. K. Transformer-based generative adversarial networks in computer vision: A comprehensive survey. *IEEE Transactions on Artificial Intelligence*, IEEE, 2024. Disponível em: <https://doi.org/10.1109/TAI.2024.3404910>.

[34] GROUP, O. R. *Datasets used in Publications*. <https://www.uel.br/grupos/orion/datasets.html>. Accessed: 2024-01-08.

[35] UNB. *DDoS Evaluation Dataset (CIC-DDoS2019)*. <https://www.unb.ca/cic/datasets/ddos-2019.html>. Accessed: 2023-04-10.

[36] UNB. *Intrusion detection evaluation dataset (CIC-IDS2017)*. <https://www.unb.ca/cic/datasets/ids-2017.html>. Accessed: 2024-05-31.

[37] 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>.

[38] 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>.

[39] KADRI, M. R. et al. Survey and classification of dos and ddos attack detection and validation approaches for iot environments. *Internet of Things*, Elsevier, p. 101021, 2023. Disponível em: <https://doi.org/10.1016/j.iot.2023.101021>.

[40] BALA, B.; BEHAL, S. Ai techniques for iot-based ddos attack detection: Taxonomies, comprehensive review and research challenges. *Computer science review*, Elsevier, v. 52, p. 100631, 2024. Disponível em: <https://doi.org/10.1016/j.cosrev.2024.100631>.

[41] PANDEY, N.; MISHRA, P. K. Devising a hybrid approach for near real-time ddos detection in iot. *Computers and Electrical Engineering*, Elsevier, v. 118, p. 109448, 2024. Disponível em: <https://doi.org/10.1016/j.compeleceng.2024.109448>.

[42] RUFFO, V. G. da S. et al. Anomaly and intrusion detection using deep learning for software-defined networks: A survey. *Expert Systems with Applications*, Elsevier, p. 124982, 2024. Disponível em: <https://doi.org/10.1016/j.eswa.2024.124982>.

[43] RUFFO, V. G. da S. et al. Generative adversarial networks to detect intrusion and anomaly in ip flow-based networks. *Future Generation Computer Systems*, Elsevier, v. 163, p. 107531, 2025. Disponível em: <https://doi.org/10.1016/j.future.2024.107531>.

[44] ZACARON, A. M. et al. Generative adversarial network models for anomaly detection in software-defined networks. *Journal of Network and Systems Management*, Springer, v. 32, n. 4, p. 93, 2024. Disponível em: <https://doi.org/10.1007/s10922-024-09867-z>.

[45] 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>.

[46] 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>.

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

[48] 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>.

[49] 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>.

[50] 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>.

[51] 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>.

[52] 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>.

[53] 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>.

[54] 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>.

[55] 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>.

[56] HNAMTE, V.; HUSSAIN, J. Dcnnbilstm: An efficient hybrid deep learning-based intrusion detection system. *Telematics and Informatics Reports*, Elsevier, v. 10, p. 100053, 2023. Disponível em: <https://doi.org/10.1016/j.teler.2023.100053>.

[57] 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>.

[58] SALAMI, A.; ANDREU-PEREZ, J.; GILLMEISTER, H. Eeg-itnet: An explainable inception temporal convolutional network for motor imagery classification. *IEEE Access*, IEEE, v. 10, p. 36672–36685, 2022. Disponível em: <https://doi.org/10.1109/ACCESS.2022.3161489>.

[59] 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>.

[60] SAEED, W.; OMLIN, C. Explainable ai (xai): A systematic meta-survey of current challenges and future opportunities. *Knowledge-based systems*, Elsevier, v. 263, p. 110273, 2023. Disponível em: <https://doi.org/10.1016/j.knosys.2023.110273>.

[61] HASSIJA, V. et al. Interpreting black-box models: a review on explainable artificial intelligence. *Cognitive Computation*, Springer, v. 16, n. 1, p. 45–74, 2024. Disponível em: <https://doi.org/10.1007/s12559-023-10179-8>.

[62] SAMED, A.; SAGIROGLU, S. Explainable artificial intelligence models in intrusion detection systems. *Engineering Applications of Artificial Intelligence*, Elsevier, v. 144, p. 110145, 2025. Disponível em: <https://doi.org/10.1016/j.engappai.2025.110145>.

[63] MOUSTAFA, N. et al. Explainable intrusion detection for cyber defences in the internet of things: Opportunities and solutions. *IEEE Communications Surveys & Tutorials*, IEEE, v. 25, n. 3, p. 1775–1807, 2023. Disponível em: <https://doi.org/10.1109/COMST.2023.3280465>.

[64] ALI, S. et al. Explainable artificial intelligence (xai): What we know and what is left to attain trustworthy artificial intelligence. *Information fusion*, Elsevier, v. 99, p. 101805, 2023. Disponível em: <https://doi.org/10.1016/j.inffus.2023.101805>.

[65] KÖK, I. et al. Explainable artificial intelligence (xai) for internet of things: a survey. *IEEE Internet of Things Journal*, IEEE, v. 10, n. 16, p. 14764–14779, 2023. Disponível em: <https://doi.org/10.1109/JIOT.2023.3287678>.

[66] LUNDBERG, S. M.; LEE, S.-I. A unified approach to interpreting model predictions. In: . Red Hook, NY, USA: Curran Associates Inc., 2017. (NIPS'17), p. 4768–4777. ISBN 9781510860964.

[67] KOMARCHESQUI, M. et al. Explainable ai feature selection in generative adversarial networks system aiming to detect ddos attacks. In: IEEE. *2024 11th International Conference on Software Defined Systems (SDS)*. 2024. p. 27–34. Disponível em: <https://doi.org/10.1109/SDS64317.2024.10883903>.

[68] SABEEL, U. et al. Unknown, atypical and polymorphic network intrusion detection: A systematic survey. *IEEE Transactions on Network and Service Management*, v. 21, n. 1, p. 1190–1212, 2024. Disponível em: <https://doi.org/10.1109/TNSM.2023.3298533>.

[69] MELIS, A. et al. A systematic literature review of offensive and defensive security solutions with software defined network. *IEEE Access*, v. 11, p. 93431–93463, 2023. Disponível em: <https://doi.org/10.1109/ACCESS.2023.3276238>.

[70] ABDULGANIYU, O. H.; TCHAKOUCHT, T. A.; SAHEED, Y. K. A systematic literature review for network intrusion detection system (ids). *International Journal of Information Security*, Springer, v. 22, n. 5, p. 1125–1162, 2023. Disponível em: <https://doi.org/10.1007/s10207-023-00682-2>.

[71] 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>.

[72] CHERIAN, M.; VARMA, S. L. Secure sdn–iot framework for ddos attack detection using deep learning and counter based approach. *Journal of Network and Systems Management*, Springer Science and Business Media LLC, v. 31, n. 3, jun. 2023. ISSN 1573-7705. Disponível em: <https://doi.org/10.1007/s10922-023-09749-w>.

[73] TAYFOUR, O. E. et al. Adapting deep learning-lstm method using optimized dataset in sdn controller for secure iot. *Soft Computing*, Springer Science and Business Media LLC, maio 2023. ISSN 1433-7479. Disponível em: <https://doi.org/10.1007/s00500-023-08348-w>.

[74] SOLTANI, M.; SIAVOSHANI, M. J.; JAHANGIR, A. H. A content-based deep intrusion detection system. *International Journal of Information Security*, Springer Science and Business Media LLC, v. 21, n. 3, p. 547–562, set. 2021. ISSN 1615-5270. Disponível em: <https://doi.org/10.1007/s10207-021-00567-2>.

[75] HOUDA, Z. A. E.; BRIK, B.; KHOUKHI, L. "why should i trust your ids?": An explainable deep learning framework for intrusion detection systems in internet of things networks. *IEEE Open Journal of the Communications Society*, IEEE, v. 3, p. 1164–1176, 2022. Disponível em: <https://doi.org/10.1109/OJCOMS.2022.3188750>.

[76] SHAJI, N. S. et al. Deep-discovery: Anomaly discovery in software-defined networks using artificial neural networks. *Computers & Security*, Elsevier, p. 103320, 2023. Disponível em: <https://doi.org/10.1016/j.cose.2023.103320>.

[77] KUMAR, C. et al. Nature-inspired intrusion detection system for protecting software-defined networks controller. *Computers & Security*, Elsevier, v. 134, p. 103438, 2023. Disponível em: <https://doi.org/10.1016/j.cose.2023.103438>.

[78] 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>.

[79] PARK, C. et al. An enhanced ai-based network intrusion detection system using generative adversarial networks. *IEEE Internet of Things Journal*, IEEE, v. 10, n. 3, p. 2330–2345, 2022. Disponível em: <https://doi.org/10.1109/JIOT.2022.3211346>.

[80] KUMAR, V.; SINHA, D. Synthetic attack data generation model applying generative adversarial network for intrusion detection. *Computers & Security*, Elsevier, v. 125, p. 103054, 2023. Disponível em: <https://doi.org/10.1016/j.cose.2022.103054>.

[81] OUALI, Y.; HUDELOT, C.; TAMI, M. An overview of deep semi-supervised learning. *arXiv preprint arXiv:2006.05278*, 2020. Disponível em: <https://doi.org/10.48550/arXiv.2006.05278>.

[82] BOPPANA, T. K.; BAGADE, P. Gan-ae: An unsupervised intrusion detection system for mqtt networks. *Engineering Applications of Artificial Intelligence*, Elsevier, v. 119, p. 105805, 2023. Disponível em: <https://doi.org/10.1016/j.engappai.2022.105805>.

[83] YAO, W.; SHI, H.; ZHAO, H. Scalable anomaly-based intrusion detection for secure internet of things using generative adversarial networks in fog environment. *Journal of Network and Computer Applications*, Elsevier, v. 214, p. 103622, 2023. Disponível em: <https://doi.org/10.1016/j.jnca.2023.103622>.

[84] LI, Z. et al. Abnormal traffic detection: Traffic feature extraction and dae-gan with efficient data augmentation. *IEEE Transactions on Reliability*, IEEE, 2022. Disponível em: <https://doi.org/10.1109/TR.2022.3204349>.

[85] ADIBAN, M.; SINISCALCHI, S. M.; SALVI, G. A step-by-step training method for multi generator gans with application to anomaly detection and cybersecurity. *Neurocomputing*, Elsevier, v. 537, p. 296–308, 2023. Disponível em: <https://doi.org/10.1016/j.neucom.2023.03.056>.

[86] XU, L. et al. Tgan-ad: Transformer-based gan for anomaly detection of time series data. *Applied Sciences*, MDPI, v. 12, n. 16, p. 8085, 2022. Disponível em: <https://doi.org/10.3390/app12168085>.

[87] MALL, R. et al. Stacking ensemble approach for ddos attack detection in software-defined cyber–physical systems. *Computers and Electrical Engineering*, Elsevier, v. 107, p. 108635, 2023. Disponível em: <https://doi.org/10.1016/j.compeleceng.2023.108635>.

[88] THAKKAR, A.; LOHIYA, R. A review on challenges and future research directions for machine learning-based intrusion detection system. *Archives of Computational Methods in Engineering*, Springer, p. 1–25, 2023. Disponível em: <https://doi.org/10.1007/s11831-023-09943-8>.

[89] 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>.

[90] HAN, J.; PEI, J.; TONG, H. *Data mining: concepts and techniques*. [S.l.]: Morgan kaufmann, 2022.

[91] 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>.

[92] CHICCO, D.; JURMAN, G. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, Springer, v. 21, p. 1–13, 2020. Disponível em: <https://doi.org/10.1186/s12864-019-6413-7>.

[93] KUMAR, S. et al. A comprehensive review of vulnerabilities and ai-enabled defense against ddos attacks for securing cloud services. *Computer Science Review*, Elsevier, v. 53, p. 100661, 2024.

[94] MANIVANNAN, D. Recent endeavors in machine learning-powered intrusion detection systems for the internet of things. *Journal of Network and Computer Applications*, Elsevier, p. 103925, 2024.

[95] CONTRIBUTORS, M. P. *Mininet: An Instant Virtual Network on your Laptop (or other PC)*. <http://mininet.org/>. Accessed: 2023-12-05.

[96] MACIÁ-FERNÁNDEZ, G. et al. Ugr '16: A new dataset for the evaluation of cyclostationarity-based network idss. *Computers & Security*, Elsevier, v. 73, p. 411–424, 2018. Disponível em: <https://doi.org/10.1016/j.cose.2017.11.004>.

[97] MAATEN, L. Van der; HINTON, G. Visualizing data using t-sne. *Journal of machine learning research*, v. 9, n. 11, 2008.

TRABALHOS PUBLICADOS PELO AUTOR

Trabalhos publicados pelo autor durante o programa.

1. **Vitor Gabriel da Silva Ruffo**, Daniel Matheus Brandão Lent, Mateus Komarchesqui, Vinícius Ferreira Schiavon, Marcos Vinicius Oliveira de Assis, Luiz Fernando Carvalho, Mario Lemes Proença Junior, **Anomaly and intrusion detection using deep learning for software-defined networks: A survey**, *Expert Systems with Applications*, 08/2024, Elsevier, Qualis periódicos 2017-2020, A1. DOI: 10.1016/j.eswa.2024.124982.

2. **da Silva Ruffo, Vitor G.**, Daniel M. Brandão Lent, Luiz F. Carvalho, Jaime Lloret, and Mario Lemes Proença Jr., **Generative adversarial networks to detect intrusion and anomaly in IP flow-based networks.**, *Future Generation Computer Systems*, 02/2025, Elsevier, Qualis periódicos 2017-2020, A1. DOI: 10.1016/j.future.2024.107531.

3. **Vitor Gabriel da Silva Ruffo**, Luiz Fernando Carvalho, Jaime Lloret, Mario Lemes Proença Junior, **f-AnoGAN for Unsupervised Attack Detection in SDN Environment**, *IEEE Transactions on Network Science and Engineering*, 04/2025, IEEE, Qualis periódicos 2017-2020, A1. DOI: 10.1109/TNSE.2025.3558936.

4. **Vitor Gabriel da Silva Ruffo**, Luiz Fernando Carvalho, Jaime Lloret, Mario Lemes Proenca, **Unsupervised *DDoS* Detection Using Entropy Features and *f-AnoGAN* in Software-Defined Networks**, *11th IEEE International Conference on Software Defined Systems*, 12/2024, IEEE. Qualis conferências 2017-2020, A4. DOI: 10.1109/SDS64317.2024.10883901.

5. Zacaron, Alexandro Marcelo, Daniel Matheus Brandão Lent, **Vitor Gabriel da Silva Ruffo**, Luiz Fernando Carvalho, and Mario Lemes Proença Jr., **Generative Adversarial Network Models for Anomaly Detection in Software-Defined Networks.**, *Journal of Network and Systems Management*, 09/2024, Springer, Qualis periódicos 2017-2020, A2. DOI: 10.1007/s10922-024-09867-z.

6. Daniel M. Brandão Lent, **Vitor G. da Silva Ruffo**, Luiz F. Carvalho, Jaime Lloret, Joel J. P. C. Rodrigues, Mario Lemes Proença Jr., **An Unsupervised Generative Adversarial Network System to Detect *DDoS* Attacks in SDN**, *IEEE Access*, 05/2024, IEEE, Qualis periódicos 2017-2020, A3. DOI: 10.1109/ACCESS.2024.3402069.

7. Mateus Komarchesqui, Daniel Matheus Brandao Lent, **Vitor Gabriel da Silva Ruffo**, Luiz Fernando Carvalho, Jaime Lloret, Mario Lemes Proenca, **Explainable AI Feature Selection in Generative Adversarial Networks System aiming to detect *DDoS* Attacks**, *11th IEEE International Conference on Software Defined Systems*, 12/2024, IEEE. Qualis conferências 2017-2020, A4. DOI: 10.1109/SDS64317.2024.10883903.