Universidade Estadual de Londrina

VINÍCIUS FERREIRA SCHIAVON

REDE ADVERSÁRIA GENERATIVA BASEADA EM TRANSFORMADOR E XAI PARA DETECÇÃO DE INTRUSÕES E ANOMALIAS EM REDES DEFINIDAS POR SOFTWARE

LONDRINA
2025

VINÍCIUS FERREIRA SCHIAVON

REDE ADVERSÁRIA GENERATIVA BASEADA EM TRANSFORMADOR E XAI PARA DETECÇÃO DE INTRUSÕES E ANOMALIAS EM REDES DEFINIDAS POR SOFTWARE

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

Orientador: Prof. Dr. Mario Lemes Proença Jr.

LONDRINA

2025

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

S329r Schiavon, Vinícius Ferreira.
Rede Adversária Generativa baseada em Transformador e XAI para Detecção de Intrusões e Anomalias em Redes Definidas por Software / Vinícius Ferreira Schiavon. - Londrina, 2025.
80 f. : il.

Orientador: Mario Lemes Proença Jr..
Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Universidade Estadual de Londrina, Centro de Ciências Exatas, Graduação em Ciência da Computação, 2025.
Inclui bibliografia.

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

CDU 519

VINÍCIUS FERREIRA SCHIAVON

REDE ADVERSÁRIA GENERATIVA BASEADA EM TRANSFORMADOR E XAI PARA DETECÇÃO DE INTRUSÕES E ANOMALIAS EM REDES DEFINIDAS POR SOFTWARE

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

BANCA EXAMINADORA

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

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

Mateus Komarchesqui
Universidade Estadual de Londrina

Londrina, 04 de fevereiro de 2025.

Dedico este trabalho à minha mãe, ao meu pai e ao meu irmão, que, apesar da distância física, sempre estiveram presentes durante todo percurso da minha graduação.

AGRADECIMENTOS

Aos meus pais, que sempre trabalharam duro para me proporcionar um futuro digno e que me inspiram diariamente a ser alguém melhor. E aos demais familiares pelo amor e pelo apoio constantes, com um agradecimento especial ao meu irmão.

Ao professor Dr. Mario Lemes Proença Jr., por todos os ensinamentos e orientações fundamentais na minha trajetória acadêmica, permitindo-me desenvolver este e outros trabalhos.

Aos meus colegas de curso, pela parceria e pelo suporte nos momentos difíceis, bem como pela amizade nos momentos de celebração.

Aos membros do grupo Orion da UEL, por sempre se prontificarem a sanar minhas dúvidas, não medindo esforços quando necessário.

Aos professores do curso de Ciência da Computação, por me concederem conhecimentos valiosos e um amadurecimento imensurável ao longo da minha graduação.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico, pelos financiamentos que possibilitam o avanço das pesquisas científicas.

Aos servidores da UEL, pelo seu trabalho essencial que garante o funcionamento de toda a universidade.

"Aquele que tem um 'porquê' para viver
pode enfrentar quase todos os 'comos'."
Friedrich Nietzsche

SCHIAVON, V. F.. **Rede Adversária Generativa baseada em Transformador e XAI para Detecção de Intrusões e Anomalias em Redes Definidas por Software**. 2025. 80f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2025.

RESUMO

A crescente complexidade e integração das redes de computadores intensificam a necessidade de mecanismos robustos para detecção e mitigação de anomalias e intrusões. Redes Definidas por Software oferecem flexibilidade e escalabilidade, mas também introduzem vulnerabilidades críticas devido à centralização do controlador. Este trabalho investiga a integração de Redes Adversárias Generativas com a arquitetura do Transformador para desenvolver um Sistema de Detecção de Intrusão em Redes não supervisionado. O modelo proposto utiliza métodos de Inteligência Artificial Explicável para interpretar as predições do sistema, promovendo maior confiança e entendimento. O estudo avalia o desempenho do modelo utilizando um conjunto de dados sintético de uma rede SDN emulada que sofre ataques de *DDoS* e *Port Scan*. O desempenho foi demarcado por meio de métricas como Acurácia, Precisão, Revocação, F1-*score*, Coeficiente de Correlação de Matthews, Área Sob a Curva e Característica de Operação do Receptor, onde o modelo apresentou valores superiores a 96% para todas elas. A Precisão do modelo foi a menor das métricas, sugerindo que ele não foi capaz de aprender totalmente a distribuição normal dos dados. O modelo foi capaz de detectar os ataques, mas apresentou instabilidade no treino, interferindo negativamente no seu desempenho.

Palavras-chave: Segurança de redes. Detecção de anomalias. Detecção de intrusões. Rede Adversária Generativa. Transformador. Inteligência Artificial Explicável.

SCHIAVON, V. F.. **Generative Adversarial Network based on Trnaformer and XAI for Intrusion and Anomaly Detection in Software Defined Netowrks**. 2025. 80p. Final Project (Bachelor of Science in Computer Science) – State University of Londrina, Londrina, 2025.

ABSTRACT

The growing complexity and integration of computer networks underscore the need for robust mechanisms to detect and mitigate anomalies and intrusions. Software-Defined Networks provide flexibility and scalability but also introduce critical vulnerabilities due to controller centralization. This study explores the integration of Generative Adversarial Networks with Transformer architecture to develop an unsupervised Intrusion Detection System. The proposed model incorporates Explainable Artificial Intelligence methods to interpret system predictions, fostering greater trust and understanding. The research evaluates the model's performance using a synthetic dataset of an emulated SDN under DDoS and Port Scan attacks. Performance was measured through metrics such as Accuracy, Precision, Recall, F1-score, Matthews Correlation Coefficient, Area Under the Curve and Receiver Operating Characteristic, where the model presented values over 96% for all of them. While the model demonstrated the ability to detect attacks, its lower precision suggests a limited capability to fully learn the normal data distribution. Additionally, training instability negatively impacted its overall performance.

Keywords: Network security. Anomaly detection. Intrusion detection. Generative Adversarial Network. Transformer. Explainable Artificial Intelligence.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Adam	*Adaptive Moment Estimation*
AI	*Artificial Intelligence*
ALE	*Accumulated Local Effects*
AUC	*Area Under the Receiver Operating Characteristic curve*
BLEU	*Bilingual Evaluation Understudy*
D	Discriminador
DDoS	*Distributed Denial of Service*
DL	*Deep Learning*
DNN	*Deep Neural Network*
FN	Falso Negativo
FP	Falso Positivo
FPR	*False Positive Rate*
G	Gerador
GAN	*Generative Adversarial Network*
Gb	Gigabit
GPT	*Generative Pre-trained Transformer*
IP	*Internet Protocol*
LIME	*Locally Interpretable Model-Agnostic Explainer*
MCC	*Matthews Correlation Coefficient*
Mb	Megabit
ML	*Machine Learning*
NIDS	*Network Intrusion Detection System*
NN	*Neural Network*
RAM	*Random Access Memory*

ReLU	*Rectified Linear Unit*
ROC	*Receiver Operating Characteristic*
SDN	*Software Defined Network*
SHAP	*Shapley Additive Explanations*
Tb	Terabit
VN	Verdadeiro Negativo
VP	Verdadeiro Positivo
VTT	*Variable Temporal Transformer*
WGAN	*Wasserstein GAN*
WGAN-GP	*Wasserstein GAN with Gradient Penalty*
XAI	*Explainable Artificial Intelligence*

LISTA DE SÍMBOLOS

$\mathbb{P}$	Distribuição de Probabilidade
$\mathbb{E}$	Valor Esperado
$\in$	Pertence
λ	Lambda
∇	Gradiente
Σ	Somatório
μ	Média
σ	Desvio-padrão

SUMÁRIO

1 INTRODUÇÃO

A utilização de aparelhos auxiliares, como celulares, computadores, sensores e tecnologias vestíveis está cada vez mais presente no dia a dia dos seres humanos. Estas ferramentas possuem acesso à Internet, possibilitando sua intercomunicação e compondo uma rede de computadores [2]. A constante integração de novos dispositivos contribui para o crescimento do volume de tráfego, exigindo melhorias na infraestrutura das redes de computadores, como o acréscimo de *switches* e roteadores. Estas alterações acabam aumentando a diversidade de componentes estruturais, dificultando sua gerência, pois cada fabricante define sua própria interface de programação [3]. Para alterar políticas de gerenciamento ou lógicas de controle, é necessário realizar mudanças em cada dispositivo que compõe a rede. Esta dificuldade demonstra a baixa escalabilidade e adaptabilidade a mudanças da arquitetura tradicional.

A arquitetura de Rede Definida por *Software* foi criada com o objetivo de solucionar essas limitações, sendo capaz de acompanhar a evolução das redes. Esse paradigma separa o plano de controle do plano de dados, representados pelo controlador e pelos *switches* e roteadores, respectivamente. O plano de dados tem o papel de encaminhar os pacotes do tráfego para seus devidos destinos, já o plano de controle fica responsável por manter toda lógica de controle [4]. Essa unificação do ponto de controle possibilita que políticas de tráfego sejam implementadas em uma única interface de programação, que se comunicará com os demais componentes por meio de um protocolo em comum (por exemplo, *OpenFlow*). Apesar da centralização proporcionar benefícios, ela também cria um ponto único de falhas, pois caso o controlador não esteja funcional, todo sistema torna-se inoperável.

Sistemas de missão crítica, como as atuais redes de computadores, requerem alta disponibilidade, resiliência e não podem sofrer interrupções nos serviços prestados. Anomalias e tentativas de intrusão podem comprometer a confidencialidade, integridade e disponibilidade dos dados, também impactando negativamente no desempenho da infraestrutura da rede [5]. Esses problemas incluem a interrupção de serviços, roubo de informações sensíveis, extorsão e outros danos causados por ataques maliciosos [6]. A crescente diversidade de formas de ataque e o aumento exponencial no volume do tráfego tornam a detecção, e possível mitigação, dessas ameaças uma prioridade [7]. A implementação de sistemas de segurança eficazes é indispensável para garantir o funcionamento adequado das redes.

Os Sistemas de Detecção de Intrusões em Redes são ferramentas essenciais para monitorar e identificar comportamento anômalos ou maliciosos no tráfego. Existem duas abordagens principais para o desenvolvimento desses sistemas: os baseados em assinaturas,

que detectam ataques conhecidos, comparando padrões de tráfego com um banco de dados pré-existente [8], e os baseados em anomalias, que identificam desvios significativos do comportamento esperado do tráfego [9]. Com o avanço dos ataques em sofisticação e volume, a aplicação de modelos de Aprendizado de Máquina, especificamente Aprendizado Profundo, na construção de sistemas baseados em anomalias tornou-se cada vez mais prevalente, permitindo a detecção de ameaças emergentes e desconhecidas com maior precisão. Esses modelos destacam-se ao trabalhar com séries temporais [10], sendo o caso do tráfego das redes.

As Redes Adversárias Generativas e os Transformadores são arquiteturas de Aprendizado Profundo que se destacaram nos últimos anos pela eficiência na resolução de suas respectivas tarefas. As Redes Adversárias aprendem distribuições complexas de dados para gerarem dados sintéticos semelhantes aos originais [11], [12], [13]. Os Transformadores revolucionaram o processamento sequencial de informações com mecanismos de Atenção [1], [14], [15]. Apesar de terem inicialmente sido desenvolvidas para outros escopos, as duas arquiteturas mostraram-se eficientes na detecção de anomalias de forma não supervisionada, principalmente por conseguirem modelar padrões de dados [16], [17]. Estas arquiteturas nem sempre geram resultados explicáveis, sendo necessário o uso de técnicas para obter interpretações sobre as predições realizadas.

A Inteligência Artificial Explicável tem como um de seus propósitos a necessidade de compreender o funcionamento interno de modelos de Inteligência Artificial, especialmente os que trabalham com dados sensíveis [18], como é o caso do tráfego da rede. Para cumprir com este objetivo, são utilizados métodos que permitem explorar como as decisões dos modelos são tomadas, revelando as influências de diferentes características nos resultados e fornecendo maior transparência no processo [19]. Essa capacidade é crucial para aumentar a confiança dos usuários, também facilitando o desenvolvimento do modelo e provendo análises científicas acerca do seu comportamento.

Este trabalho está estruturado para abordar de maneira sistemática o desenvolvimento e avaliação de um sistema de detecção de intrusões e anomalias baseado em Redes Adversárias Generativas, Transformadores e Inteligência Artificial Explicável. O Capítulo 2 apresenta os fundamentos teóricos necessários, incluindo Redes Definidas por Software e Aprendizado de Máquina. No Capítulo 3, a arquitetura de Redes Adversárias Generativas é explorada em detalhe, enquanto o Capítulo 4 se concentra nos Transformadores. A Inteligência Artificial Explicável e seus métodos são discutidos no Capítulo 5. Os Capítulos 6 e 7 abrangem os trabalhos relacionados e o estudo de caso, respectivamente, chegando na análise dos resultados e nas conclusões, apresentadas no Capítulo 8.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Redes Definidas por Software

Redes de computadores desempenham um papel fundamental na comunicação global. Elas são compostas de dispositivos interconectados que permitem o compartilhamento de dados e recursos computacionais [5]. Com a evolução tecnológica, as redes cresceram em velocidade e complexidade, possibilitando aplicações cada vez mais sofisticadas. O tráfego da rede é composto por pacotes que levam informações de um ponto a outro seguindo um protocolo pré-definido. Para sair do seu ponto de origem e alcançar o seu ponto de destino, os pacotes são roteados por meio dos nós que compõem a infraestrutura de rede [20].

A Figura 1 apresenta a arquitetura tradicional das redes de computadores. Cada componente da infraestrutura (*switches* ou roteadores) possui dois planos acoplados: o de controle e o de dados. O plano de dados é responsável pelo encaminhamento de pacotes com base em regras já definidas no seu respectivo plano de controle [21]. Também há o plano de aplicação, onde se localizam as aplicações que usufruirão das funcionalidades oferecidas pela rede, como ferramentas de análise de tráfego, sistemas de segurança (como *firewalls* e de detecção de intrusões e anomalias) e aplicativos de gerenciamento [22]. Essas aplicações são cruciais para garantir a segurança e funcionamento de toda a arquitetura.

Quando se deseja realizar alterações nas políticas de gerência em uma rede tradicional, é necessário alterar cada plano de controle existente. Esta atividade é complexa e custosa em tempo, pois dispositivos oriundos de fabricantes diferentes possuem linguagens de programação distintas [23]. Esta dificuldade demonstra que a arquitetura tradicional é inflexível e dificulta a escalabilidade.

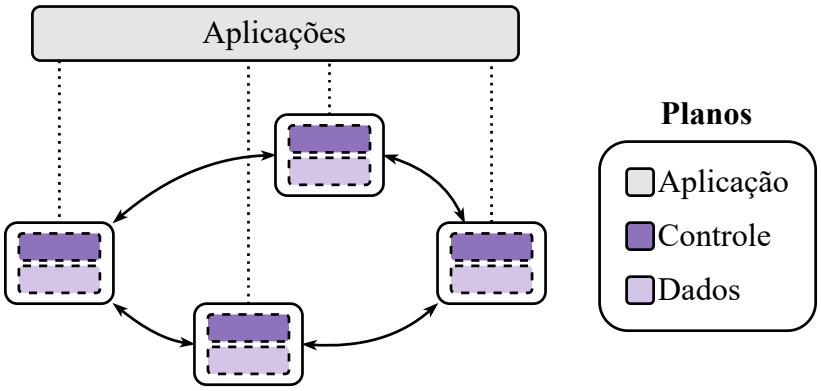

Figura 1 – Arquitetura tradicional de redes. Fonte: elaboração própria.

A Figura 2 exibe a arquitetura de Redes Definidas por *Software* (SDN, do inglês

Software Defined Network). A mudança está na adição do controlador, um novo componente logicamente centralizado. Ele mantém as regras da rede (plano de controle) e possui duas interfaces: a *northbound* e a *southbound* [24]. A interface *northbound* tem o propósito de se comunicar com o plano de aplicações, que mantém as mesmas funcionalidades que possuia na arquitetura tradicional. Já a interface *southbound* tem como principal função o controle e gerenciamento do plano de dados. Esta interface, utilizando protocolos de comunicação (como *OpenFlow*), age como intermediária entre o plano de controle e o de dados [2]. Os *switches* e roteadores passam a ser responsáveis apenas pelo encaminhamento de pacotes (plano de dados), já que o controlador mantém as políticas de tráfego.

A centralização oferecida pelo controlador torna a rede mais flexível e escalável, pois as mudanças de políticas e a adição de novos componentes são gerenciadas em um único ponto. Esta característica contribui para diminuição, em relação à arquitetura tradicional, da complexidade e custo em tempo das alterações necessárias [25]. O controlador também apresenta como vantagem a capacidade de extrair estatísticas sobre o tráfego, podendo ser utilizadas para auxiliar na tomada de decisões e gerência da rede [26].

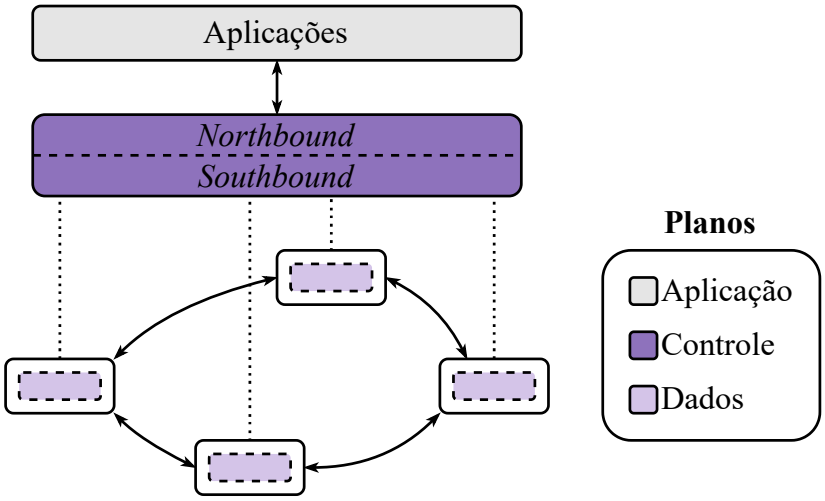

Figura 2 – Arquitetura SDN. Fonte: elaboração própria.

Apesar dos benefícios, a arquitetura SDN possui desvantagens. O funcionamento adequado da rede depende do controlador, e caso ele não esteja funcional, o restante da rede ficará disfuncional. Esta vulnerabilidade pode ser alvo de ataques de rede, como negação de serviço [27]. Apesar da existência desta vulnerabilidade, a visão global da rede proporcionada pelo controlador auxilia na detecção de possíveis ameaças [28]. A comunidade científica tem estudado e desenvolvido soluções que utilizem as vantagens do controlador para minimizar danos causados por intrusões e anomalias.

2.2 Intrusões e Anomalias em Redes de Computadores

Considera-se anomalia todo registro de tráfego cujo comportamento destoa, acima de uma determinada tolerância, do esperado. Anomalias podem ocorrer por falhas humanas, problemas técnicos na rede ou deliberadamente por meio de atividades maliciosas (intrusões) [4]. O padrão de comportamento esperado, também chamado de *baseline*, é resultado da análise comportamental feita sobre o histórico de tráfego da rede [29].

Intrusões de rede são um exemplo de atividade maliciosa realizada por meio de ataques que buscam penetrar na rede sem serem notados [30]. Estes ataques possuem diversos objetivos, como roubo de dados, injeção de dados, interrupção de serviços, extorsão financeira e roubo do poder de processamento de um computador. Quando a eficiência do ataque depende do volume de pacotes enviados a rede, ele se enquadra como volumétrico [31]. Os ataques abordados nesse trabalho serão [3]:

- Varredura de Portas (*Port Scan*): é uma técnica utilizada para identificar portas abertas em um sistema alvo. O atacante envia pacotes para uma série de portas, analisando as respostas para determinar quais estão ativas e potencialmente vulneráveis. Pode ser usado legitimamente por administradores de rede ou como um precursor de outros ataques.
- Negação de Serviço Distribuído (*DDoS*, do inglês *Distributed Denial of Service*): visa sobrecarregar um sistema ou rede com um volume massivo de tráfego proveniente de múltiplas fontes simultaneamente. O objetivo é esgotar os recursos e interromper o serviço do alvo, tornando-o inacessível para usuário legítimos.

Nos últimos 40 anos, a velocidade da internet cresceu de 10 Mb/s para 800 Gb/s, apresentando um crescimento de mais de 30% ao ano [32]. Especula-se que em menos de 10 anos o valor passe a ser de 1,6 Tb/s, já existindo testes laboratoriais que chegaram a 402 Tb/s [33]. O aumento da velocidade das redes acaba potencializando alguns ataques, já que mais pacotes podem ser enviados por segundo [3]. Para evitar que essa potencialização cause mais danos, é necessário que a detecção das intrusões seja feita o mais rápido possível [5]. Uma solução que atende esse requisito é o Sistema de Detecção de Intrusão em Redes de Computadores.

2.3 Sistema de Detecção de Intrusão em Redes de Computadores

A detecção rápida e precisa de intrusões e anomalias em redes de computadores é essencial para prevenir danos significativos à infraestrutura e aos dados. Com o aumento da sofisticação dos ataques, a implementação de sistemas robustos de detecção tornou-se

uma necessidade primordial para manter a confidencialidade, integridade e disponibilidade das redes [34].

Os Sistemas de Detecção de Intrusão de Redes (NIDS, do inglês *Network Intrusion Detection System*) são uma possível solução que busca suprir essas necessidades. Operando como uma segunda linha de defesa após o *firewall*, os NIDS monitoram continuamente o tráfego da rede, analisando padrões e comportamentos. Quando uma anomalia é detectada, o sistema alerta o gerente de rede, para que ele possa tomar alguma medida de mitigação [7]. A Figura 3 demonstra o funcionamento de um NIDS genérico, onde ele recebe como entrada uma amostra do tráfego a cada período t de tempo e classifica aquela instância como normal ou anômala, alertando o gerente, se necessário.

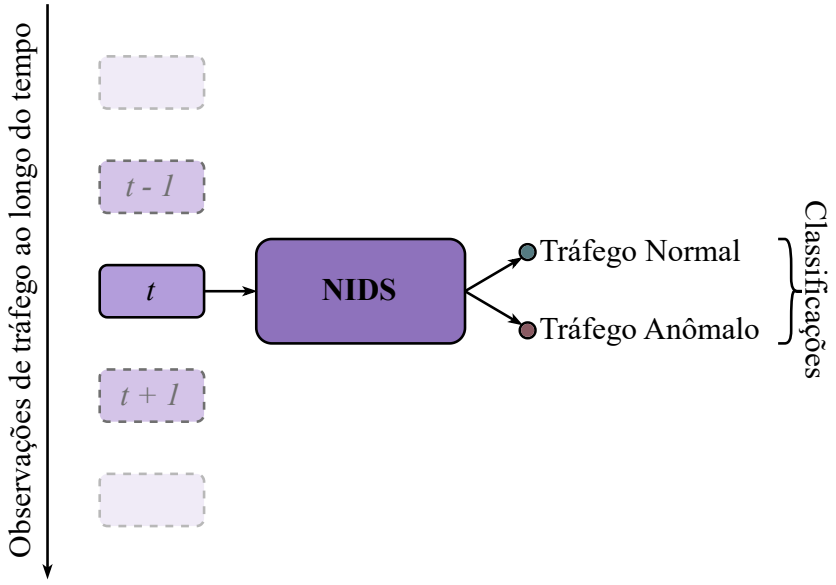

Figura 3 – Funcionamento de um NIDS. Fonte: elaboração própria.

A eficácia de um NIDS é inversamente proporcional ao seu tempo de resposta. Considerando a evolução exponencial da velocidade das redes, que atualmente chega a 400 Gigabits/s [32], o intervalo de análise torna-se um fator crítico [35]. Um NIDS deve ser capaz de processar e analisar o tráfego com a menor latência possível, para evitar que ataques causem danos antes de sua detecção [36]. Neste trabalho, o intervalo de análise será de 1 segundo.

Existem diferentes estratégias para implementação de NIDS. Uma das abordagens é a baseada em assinatura, que utiliza um banco de dados de padrões de ataques conhecidos. Este método compara o tráfego de rede com as assinaturas pré-definidas, sendo capaz de detectar ameaças conhecidas, mas limitando-se na identificação de novos tipos de ataques [8], [37]. Outra estratégia é a baseada em anomalia, que opera estabelecendo um perfil de comportamento normal da rede e identificando desvios significativos desse padrão. Esta abordagem apresenta a vantagem de poder detectar ameaças desconhecidas

ou emergentes. Já a desvantagem é que ela pode gerar mais falsos positivos do que o método baseado em assinatura, pois nem toda alteração na rede é uma intrusão [9], [38].

O campo de pesquisa em NIDS possui interesse da comunidade científica, com esforços contínuos para aprimorar a eficiência desses sistemas. É possível observar um exemplo desse esforço no grupo de pesquisa Orion da Universidade Estadual de Londrina, contribuindo desde 2002 para o avanço da ciência nesta área. Os pesquisadores do grupo já publicaram tanto revisões sistemáticas [2], [4] quanto abordagens práticas na área de NIDS [26], [22], [39], [40], [5], [41], [42], [43], [44]. Dentre os trabalhos produzidos por este grupo, é possível citar Ruffo *et al.* (2024) [2], onde os autores fazem uma pesquisa sobre os artigos desta área no período entre 2021 e 2023 para responder seis perguntas frequentes no desenvolvimento de um NIDS. Outro trabalho do grupo é o de Lent *et al.* [9], que constrói um NIDS baseado em anomalias não supervisionado, utilizando a arquitetura de Rede Adversária Generativa baseada em Unidade Recorrente Fechada. Já o trabalho de Zacaron *et al.* [45] realiza um comparativo entre diferentes variações da arquitetura de Rede Adversária Generativa, comparando os resultados a fim de ranquear cada variação. Ruffo *et al.* (2025) [7] realizaram um comparativo entre o uso de diferentes modelos de Aprendizado Profundo na arquitetura de Rede Adversária Generativa. Em geral, os trabalhos do grupo demonstram que modelos de Aprendizado de Máquina, especificamente Aprendizado Profundo, são utilizados para implementar NIDS.

2.4 Aprendizado de Máquina

O Aprendizado de Máquina (ML, do inglês *Machine Learning*) é uma subárea da Inteligência Artificial (AI, do inglês *Artificial Intelligence*). Os estudos dessa área focam em maneiras de um computador simular a capacidade de resolução de problemas e a inteligência humana [46]. A abordagem da subárea de ML é utilizar dados para treinar um modelo, tornando-o capaz de resolver um conjunto de problemas relacionados aos dados. A hierarquia das subáreas de AI que serão expostas nas próximas subseções pode ser vista na Figura 4.

Um modelo de ML tem como objetivo performar tarefas sobre um conjunto de dados. Por exemplo, é possível desenvolver um modelo que faça previsões sobre informações que o alimentam, diferentes das que foram usadas para treiná-lo [47]. Para cumprir este objetivo, espera-se que o modelo, durante o treinamento, seja ajustado automaticamente e aprenda o padrão dos conjuntos utilizados [48]. Modelos de ML suprem a necessidade de análise e interpretação sobre dados, pois são capazes de aprender padrões e relações que possivelmente não seriam encontrados por métodos tradicionais.

Figura 4 – Relação entre as subáreas de Inteligência Artificial. Fonte: elaboração própria.

2.4.1 Abordagens de Aprendizagem

A aprendizagem de modelos de ML pode ser separada em três principais abordagens, sendo:

- **Reforço** [21]: o modelo interage com um ambiente, sendo recompensado quando executa a ação correta ou penalizado quando erra. O objetivo é aprender uma política de decisão que maximize as recompensas acumuladas ao longo do treinamento. Esta abordagem é utilizada em problemas de tomada de decisão.
- **Supervisionada** [28]: cada amostra do conjunto de dados possui seu respectivo rótulo, capaz de descrevê-la. O objetivo do modelo que usa essa abordagem é aprender uma função que mapeie qualquer amostra do conjunto de dados para um rótulo próximo do original. Este tipo de aprendizagem é normalmente utilizado para resolução de problemas de classificação e regressão.
- **Não supervisionada** [27]: o conjunto de dados não possui rótulos. O modelo que emprega este tipo de aprendizagem tem como objetivo aprender padrões e estruturas que estão intrínsecos nos dados. Os usos mais comuns desta aprendizagem são para resolução de problemas de clusterização, redução de dimensionalidade e classificação.

A aprendizagem não supervisionada é vantajosa quando aplicada em cenários reais, uma vez que a tarefa de rotulação é custosa e os dados normalmente não são coletados com rótulos. Nas aplicações que empregam este tipo de aprendizagem, os dados poderão ser utilizados de maneira mais próxima ao que são encontrados no meio, não sendo necessária a etapa de rotulação [9]. O modelo desenvolvido neste trabalho será não supervisionado.

2.4.2 Rede Neural

A Rede Neural (NN, do inglês *Neural Network*) é uma especialização de ML inspirada no cérebro humano que utiliza dois principais elementos para efetuar cálculos: os neurônios e suas interconexões [49]. Os neurônios são dispostos em camadas sequenciais, e, como pode ser observado na Figura 5, um neurônio só pode ter ligação com outro se eles estiverem em camadas adjacentes. A primeira camada, chamada de camada de entrada, é responsável por receber as características de uma amostra do conjunto de dados. A última camada, chamada de camada de saída, exibe o resultado dos cálculos da NN. Entre a camada de entrada e a de saída existe ao menos uma camada oculta. A combinação de número de camadas e de neurônios em cada camada define a arquitetura de uma NN, sendo possível construir inúmeras variações ao alterar esses valores.

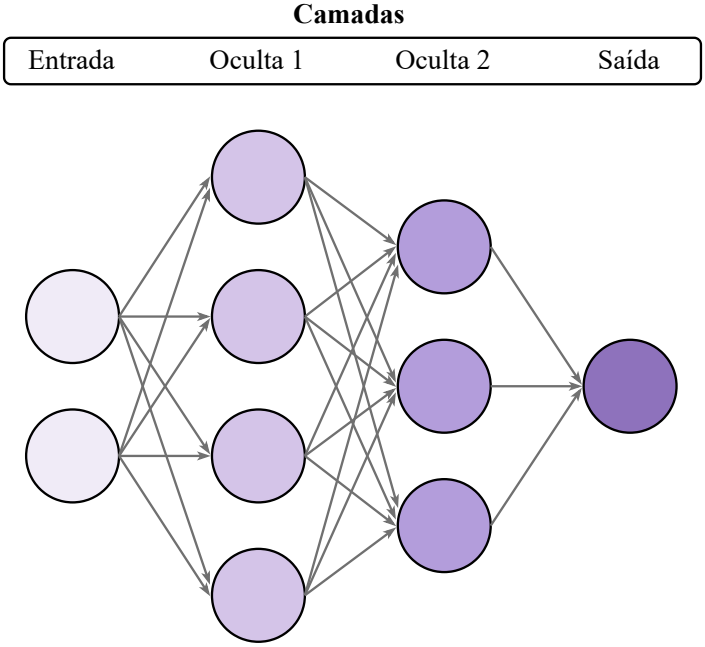

Figura 5 – Exemplo de Rede Neural. Fonte: elaboração própria.

2.4.3 Aprendizado Profundo

O Aprendizado Profundo (DL, do inglês *Deep Learning*) é uma especialização de NN, cuja diferença está relacionada ao número de camadas ocultas. A NN que possui múltiplas camadas ocultas se enquadra na subárea de DL e recebe o nome de Rede Neural Profunda (DNN, do inglês *Deep Neural Network*). Não há uma definição teórica que estabeleça um número exato de camadas ocultas que difere uma NN de uma DNN. Existe uma convenção que duas, ou mais, camadas ocultas enquadram a NN como Profunda, já apenas uma camada oculta, como Rasa [50]. Por possuir múltiplas camadas, uma DNN consegue modelar funções e resolver problemas mais complexos, porém é mais custosa computacionalmente e necessita de mais dados de treinamento [51].

Modelos de DNN se mostram eficazes no tratamento de séries temporais, como é o caso do tráfego da rede [52]. A estrutura em camadas permite a extração de características dos dados, capturando desde padrões simples até estruturas mais complexas. Esta característica possibilita que o modelo aprenda o comportamento esperado da rede. Espera-se que a presença de ataques deixe o comportamento da rede fora do padrão, possibilitando que o modelo detecte a anomalia [10].

As DNNs possuem diferentes arquiteturas que podem ser classificas em discriminativas, generativas e híbridas. Um modelo discriminativo tem a capacidade de, baseado nas características da amostra analisada, inferir uma classificação [53]. Já um modelo generativo é capaz de, baseado nas características da observação e possivelmente sua classificação, produzir uma nova amostra semelhante à classe ou dados de entrada [54]. Um modelo híbrido combina aspectos de ambas as arquiteturas [11], sendo o caso do modelo proposto neste trabalho. Para que o modelo de cada arquitetura consiga desempenhar seu propósito, é necessário passar por um processo de treinamento com um conjunto de dados que represente o problema que se busca solucionar.

2.4.4 Treinamento, Hiperparâmetros e sua Otimização

Antes de um modelo ser capaz de performar tarefas sobre um conjunto de dados, ele precisa passar pelas etapas de treinamento e teste [55]. O conjunto de dados utilizado é dividido em três partes: treino, validação e teste. O conjunto de treino é utilizado durante a etapa de treinamento para a aprendizagem do modelo. O uso do conjunto de validação possibilita a otimização de hiperparâmetros e o acompanhamento do desenvolvimento do modelo durante o treinamento [3]. Os dados de teste são utilizados no modelo já finalizado para verificar seu desempenho.

A aprendizagem de uma rede neural ocorre por meio do ajuste iterativo de suas variáveis internas, chamadas de pesos e vieses, que inicialmente são definidos com números aleatórios ou seguindo alguma distribuição estatística. Durante o treinamento, os dados de entrada são propagados pela rede produzindo um resultado, que será comparado com a saída esperada. A função de perda quantifica a diferença entre a saída real e a prevista, calculando o erro do modelo. Para minimizar o erro do modelo, um algoritmo de otimização, como o Gradiente Descendente, é utilizada para ajustar os pesos e vieses. Este processo, chamado de retropropagação, é repetido por um número pré-definido de vezes, com o objetivo de encontrar os parâmetros que minimizam a função de perda [51].

Existem variáveis que devem ser definidas previamente e não podem ser aprendidas por meio dos dados, os chamados hiperparâmetros. Eles estão diretamente relacionados com o comportamento e desempenho do modelo, podendo ser categorizados em gerais e específicos. Os hiperparâmetros específicos estão relacionados a arquiteturas ou algoritmos próprios, já os gerais são aplicáveis a uma maior diversidade de modelos de DL. Os

hiperparâmetros gerais abordados neste trabalho são [2]:

- **Número de épocas**: quantidade de vezes que o modelo percorre todo o conjunto de dados de treinamento, influenciando o tempo de aprendizagem. Um número excessivo de épocas pode causar *overfitting*, que é quando o modelo se ajusta aos dados já conhecidos, mas não a dados novos.
- **Tamanho de lote**: número de amostras processadas antes da atualização dos parâmetros do modelo, afetando a velocidade de convergência. Em cada época, antes dos dados serem divididos em relação a esse tamanho, eles podem ser embaralhados.
- **Número de camadas e neurônios por camada**: define a arquitetura da NN, impactando sua capacidade de aprender padrões complexos. O uso de valores acima dos necessários para solucionar o problema pode aumentar a complexidade do modelo e seu tempo de processamento, possibilitando o *overfitting*. Já o uso de valores abaixo dos adequados para solucionar o problema pode impedir que o modelo convirja, ocasionando o *underfitting* (processo oposto ao *overfitting*).
- **Taxa de aprendizagem**: controla a magnitude das alterações feitas ao fim de cada execução do algoritmo de otimização, influenciando na velocidade e estabilidade da aprendizagem.
- **Taxa de abandono**: percentual de neurônios aleatoriamente desativados durante o treinamento, ajudando a prevenir *overfitting* e regulando sua aprendizagem.
- **Função de ativação**: determina como o sinal de saída de um neurônio é calculado, afetando a capacidade da rede de aprender relações lineares ou não-lineares, dependendo da função utilizada.
- **Função de perda**: mede a diferença entre as previsões do modelo e os valores esperados, guiando o processo de otimização.
- **Algoritmo de otimização**: utilizado para ajustar os pesos e vieses da rede, visando minimizar a função de perda.

A otimização de hiperparâmetros é crucial para o modelo convergir, pois influencia diretamente sua capacidade de aprendizagem, generalização e eficiência computacional [2]. Existem alguns métodos de otimização, como busca em grade, busca aleatória e Otimização Bayesiana. Em relação à quantidade de combinações de hiperparâmetros, os dois primeiros métodos tendem a ser mais custosos computacionalmente do que o último. A Otimização Bayesiana também pode indicar valores de hiperparâmetros mais precisos que os outros métodos citados [56].

A Otimização Bayesiana utiliza um modelo probabilístico para modelar a relação entre os hiperparâmetros e o desempenho do modelo. O algoritmo desta otimização opera iterativamente na busca de valores ótimos dentro de intervalos pré-definidos para cada hiperparâmetro. Para direcionar a seleção de novos conjuntos de hiperparâmetros a serem testados, este método utiliza dados de avaliações anteriores. A cada iteração, o algoritmo equilibra a exploração de regiões desconhecidas do espaço de hiperparâmetros com a exploração de áreas promissoras já identificadas. Isso permite uma busca mais eficiente e direcionada, frequentemente levando a melhores resultados com menos avaliações de modelo em comparação com os outros métodos citados. A Otimização Bayesiana se torna valiosa em casos onde a avaliação do modelo é custosa, como nos casos da NN [57]. Para o algoritmo de otimização saber qual resultado foi melhor ou pior, são usadas métricas de avaliação.

2.4.5 Métricas de Avaliação

Um dos desafios na área de ML é definir um critério de avaliação e comparação que possa ser usado de maneira ampla em diferentes soluções. Em modelos de regressão a avaliação é feita ao se calcular a distância entre os valores obtidos e os valores esperados [58]. Já em modelos de classificação, utilizam-se métricas que apontam a assertividade do modelo de modo geral ou em relação a cada classe [2]. O modelo implementado neste trabalho empregará a classificação binária, onde a saída dele pode ser positiva, ao identificar uma anomalia no tráfego, ou negativa, quando o tráfego é considerado normal.

A matriz de confusão permite a obtenção de métricas primitivas que serão utilizadas no cálculo de outras métricas com propósitos específicos. As linhas dessa matriz representam as classificações obtidas em cada amostra, já as colunas representam as classificações reais. A diagonal principal representa os resultados onde o modelo classificou corretamente as amostras, já a diagonal secundária representa os erros do modelo [59]. A Figura 6 demonstra como fica disposta a matriz de confusão no caso de uma classificação binária, possuindo quatro quadrantes:

- **Verdadeiro Positivo** (VP): o modelo identifica amostras anormais como anômalas.
- **Verdadeiro Negativo** (VN): o modelo identifica amostras legítimas como normais.
- **Falso Positivo** (FP): o modelo identifica amostras legítimas como anômalas.
- **Falso Negativo** (FN): o modelo identifica amostras anormais como normais.

Os valores obtidos na matriz de confusão serão utilizados para calcular métricas que trazem diferentes percepções sobre o desempenho do modelo avaliado, sendo elas [2]:

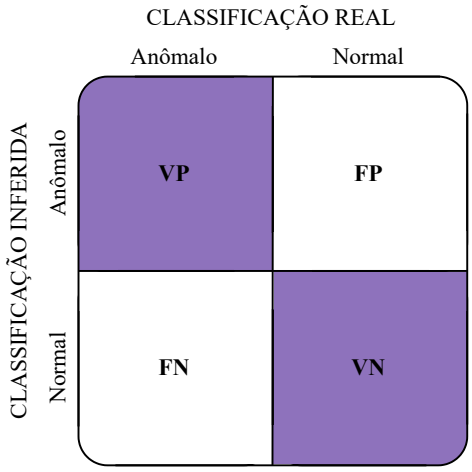

Figura 6 – Matriz de confusão para classificação binária. Fonte: elaboração própria.

- **Acurácia** (Equação 2.1): quantifica a proporção de amostras classificadas corretamente. Não recomendada caso o conjunto de dados tenha o número de amostras por classe significativamente desbalanceado, pois um modelo que classifica todas as amostras para a mesma classe pode obter alta Acurácia.

$$Acurácia = \frac{VP + VN}{VP + FP + VN + FN} \tag{2.1}$$

- **Precisão** (Equação 2.2): quantifica a proporção de predições anômalas corretas. Altos valores de Precisão indicam menos FP.

$$Precisão = \frac{VP}{VP + FP} \tag{2.2}$$

- **Revocação** (Equação 2.3): quantifica a proporção de amostras anormais preditas como anômalas. Altos valores de Revocação indicam menos FN.

$$Revocação = \frac{VP}{VP + FN} \tag{2.3}$$

- **F1-*score*** (Equação 2.4): média harmônica entre Precisão e Revocação. Oferece uma medida balanceada entre Precisão e Revocação, sendo recomendada caso o conjunto de dados esteja desbalanceado, pois caso uma das métricas da média harmônica esteja alta e a outra baixa, o valor da F1-*score* será mediano. O equilíbrio entre essas duas métricas reflete a capacidade do modelo detectar elementos relevantes (Revocação) e a assertividade dessas detecções (Precisão).

$$F1 - score = 2 \times \frac{Precisão \times Revocação}{Precisão + Revocação} \tag{2.4}$$

- **Coeficiente de Correlação de Matthews** (Equação 2.5): MCC, do inglês *Matthews Correlation Coefficient*, considera todos os quadrantes da matriz de confusão. O intervalo de possíveis valores de saída está entre -1 e 1, onde -1 representa predições

completamente invertidas, 0 indica que o modelo tem predições aleatórias, e 1 que o modelo desempenha bem nos quatro valores da matriz de confusão.

$$MCC = \frac{VP \times VN - FP \times FN}{\sqrt{(VP + FP)(VP + FN)(VN + FP)(VN + FN)}} \tag{2.5}$$

- **Taxa de Falso Positivo** (Equação 2.6): FPR, do inglês *False Positive Rate*, quantifica a proporção de amostras normais classificadas como anômalas. Baixos valores de FPR indicam menos FP.

$$TFP = \frac{FP}{FP + VN} \tag{2.6}$$

- **Característica de Operação do Receptor** (Figura 2.4.5): ROC, do inglês *Receiver Operating Characteristic*, oferece um gráfico que representa o equilíbrio entre Revocação e FPR. Cada ponto da curva representa o valor dessas métricas para um determinado limiar, sendo possível visualizar qual deles apresenta um melhor equilíbrio entre FN e FP.

- **Área Sob a Curva ROC**: AUROC, do inglês *Area Under the Receiver Operating Characteristic curve*, quantifica as informações exibidas pela curva ROC, medindo a área abaixo da curva. Valores AUROC próximos a 1 indicam que o modelo consegue diferenciar classes positivas de negativas.

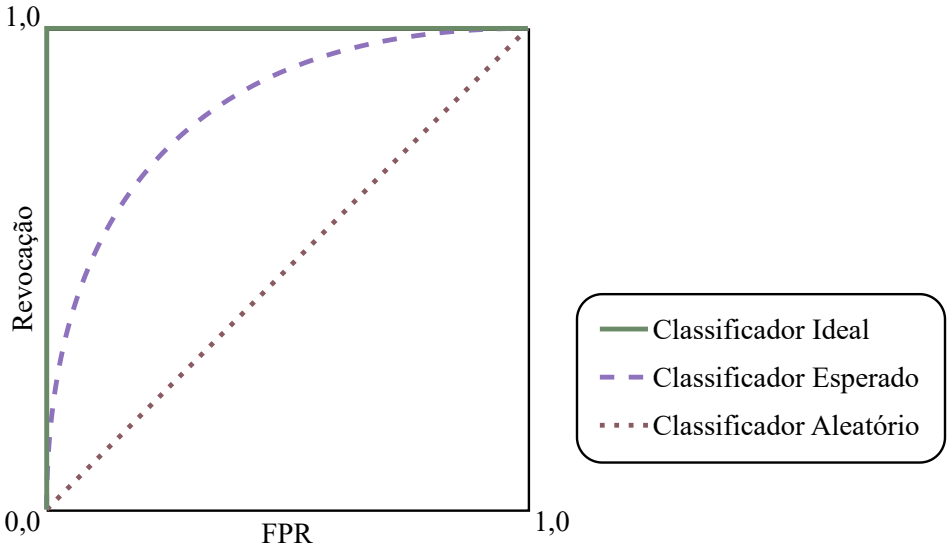

Figura 7 – Curva ROC. Fonte: elaboração própria.

Existem diferentes modelos e arquiteturas de NN com diferentes propósitos. A arquitetura de Rede Adversária Generativa atende as especificações antes ditas, sendo de DL, discriminativa e generativa, não supervisionada e capaz de processar dados a cada segundo. NIDS que utilizaram esta arquitetura, conseguiram detectar ataques de *DDoS* e *Port Scan*, obtendo métricas próximas a outros modelos de artigos científicos [16].

3 REDE ADVERSÁRIA GENERATIVA

Proposta por Goodfellow *et al.* [11], a Rede Adversária Generativa (GAN, do inglês *Generative Adversarial Network*) é uma arquitetura não supervisionada baseada na Teoria dos Jogos que implementa dois modelos de NN competindo entre si. O Gerador (G) objetiva aprender a distribuição dos dados de treinamento ($\mathbb{P}_d$) para, a partir de um vetor de ruído aleatório (z), ser capaz de gerar amostras sintéticas ($G(z)$) semelhantes às do treinamento (x). Este vetor é composto por valores que seguem uma distribuição bem definida, como a uniforme ou Gaussiana, por exemplo. Ele é extraído do chamado espaço latente, que representa a compressão das características dos dados. O Discriminador (D) tem como objetivo classificar cada amostra de entrada como pertencente ao conjunto de dados ou sintética. A disposição dos elementos da arquitetura pode ser vista na Figura 8.

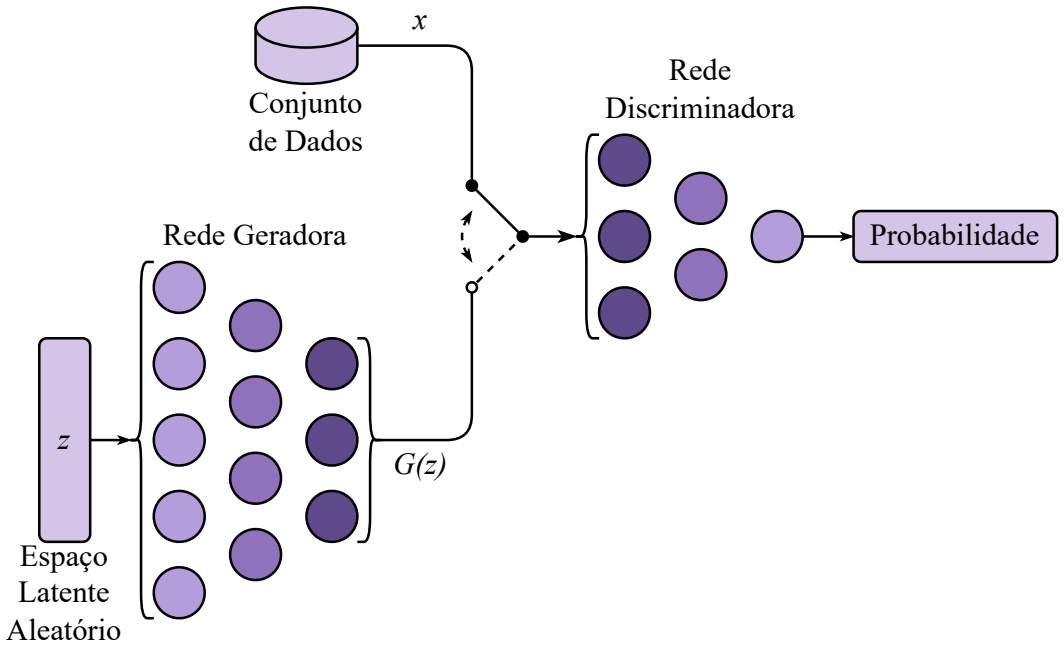

Figura 8 – Arquitetura GAN. Fonte: elaboração própria.

A característica adversária se dá no treinamento, onde G busca enganar D, que por sua vez tenta saber a origem da amostra analisada. Este processo torna os modelos progressivamente melhores na execução de suas respectivas tarefas. O objetivo da arquitetura é atingir o Equilíbrio de Nash, onde são encontrados os pesos ótimos de cada rede, a fim de minimizar a função de custo de G e maximizar a de D [60]. Quando este equilíbrio é encontrado, obtém-se um G que aprendeu a distribuição dos dados e pode gerar exemplos sintéticos semelhantes aos reais, enganando D. Também obtém-se um D capaz de diferenciar amostras oriundas do conjunto de dados ou não.

3.1 Aprendizagem

Os pesos das NN de G e D são inicializados com valores aleatórios. O treinamento será realizado em um número pré-definido de iterações, chamadas de épocas. Em cada época, a sequência de passos de treinamento será executado, utilizando o algoritmo de gradiente descendente para atualizar os pesos das NN [11]. Esta atualização ocorre para que os objetivos de cada rede sejam cumpridos, sendo o de G gerar amostras semelhantes às reais, e o de D discriminar se as amostras analisadas são reais ou geradas.

O primeiro passo do treinamento consiste em fornecer o espaço latente z para G, que utilizará seus pesos e conexões para produzir uma nova amostra de dado sintética. Logo em seguida, D receberá tanto amostras sintéticas quanto reais. Seu objetivo é indicar, por meio de uma saída entre 0 e 1, a probabilidade da amostra analisada ser real (1) ou sintética (0). D tem seus pesos atualizados com relação às classificações corretas, sendo recompensado por elas. Na próxima etapa, G recebe valores aleatórios do espaço latente para transformá-los em amostras sintéticas. G será recompensado com base nos casos onde D classifica as amostras sintéticas como reais. Ao final das épocas de treinamento, espera-se que a arquitetura esteja suficientemente próxima do Equilíbrio de Nash.

O objetivo do jogo disputado entre G e D na GAN tradicional (*Vanilla* GAN) pode ser definido pela função [11]:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim\mathbb{P}_r}[\log(D(x))] + \mathbb{E}_{\tilde{x}\sim\mathbb{P}_G}[\log(1-D(\tilde{x}))], \tag{3.1}$$

onde $\mathbb{P}_r$ representa a distribuição dos dados reais e $\mathbb{P}_G$ a distribuição do modelo implicitamente definida por $\tilde{x}=G(z)$, $z \sim \mathbb{P}_z$ (a entrada z inserida no modelo G é amostrada de alguma distribuição bem definida, como a uniforme ou Gaussiana, representada por $\mathbb{P}_z$).

O objetivo de G é minimizar o valor da função, fazendo com que D avalie todas as amostras como reais. O objetivo de D é maximizar a função, avaliando todas as amostras oriundas de G como sintéticas [11]. Esta combinação de características se enquadra como um jogo de soma-zero (*zero-sum*), onde o ganho de uma das partes necessariamente implica na perda da outra [61]. Por tratar-se da aprendizagem adversária entre dois modelos, um deles aprender seu objetivo mais rápido que o outro pode implicar em problemas, que serão melhor abordados na próxima seção.

3.2 Problemas

A característica adversária é vantajosa por empregar a aprendizagem não supervisionada. Entretanto, pela arquitetura trabalhar com duas NN, é um desafio balancear

sua aprendizagem, para que ambas aprendam em velocidades semelhantes. Alguns dos problemas causados por esse desafio são [62]:

- **Não convergência** [12]: para que o modelo GAN seja capaz de atingir o Equilíbrio de Nash, é necessário que, durante o treinamento, G e D não destoem em relação à velocidade de aprendizagem. Garantir esta sincronia entre duas NN é um desafio e algumas estratégias são empregadas, como: adicionar ruído às entradas de D ou penalizar os pesos de D. Outra alternativa que recebeu interesse da comunidade científica é estudo de novas funções de custo para serem utilizadas em G e D, criando variações da arquitetura tradicional.
- **Colapso de modo (*mode collapse*)** [13]: ocorre quando G não aprende toda distribuição dos dados de treinamento, sendo apenas capaz de gerar amostras semelhantes a um subconjunto deles. No caso deste trabalho, seria como se G criasse amostras de apenas um período de horários do dia, e não dele todo. Este problema pode ocorrer pela constante busca de G por uma saída que seja mais convincente para D. Uma solução estudada é a troca da função de divergência utilizada para calcular a distância entre as saídas reais e sintéticas.
- **Gradientes desvanecentes (*vanishing gradients*)** [63]: quando D aprende mais rápido que G, ele será menos enganado, diminuindo o retorno para G na sua atualização de pesos e dificultando sua aprendizagem. Duas das soluções adotadas são a adição de ruídos nos dados de treinamento de D e o descarte das saídas de alguns dos neurônios de D, para desacelerar sua aprendizagem.

Existe uma constante busca por melhorias na arquitetura GAN. Variantes como a GAN Condicional, GAN Convolucional Profunda ou GAN de Ciclo são feitas pensadas para um escopo específico de problemas [60]. Uma das variações que obteve destaque é a que combina a Distância de Wasserstein com a Penalidade de Gradiente. Isto ocorre, pois estas alterações não foram propostas para um tipo específico de problemas, elas foram pensadas para trazer melhorias gerais para a arquitetura, proporcionando um comportamento mais estável.

3.2.1 Wasserstein GAN com Penalidade de Gradiente

Proposta por Gulranjani *et al.* [64], a Wasserstein GAN com Penalidade de Gradiente (WGAN-GP) tem como base a Wasserstein GAN (WGAN) proposta por Arjovsky *et al.* [65]. A mudança implementada na WGAN é a alteração da função que calcula a distância entre os resultados obtidos e os esperados. A *Vanilla* GAN utiliza a divergência de Jensen-Shannon para calcular esta distância, um método que não é contínuo para todos os casos. Esta característica pode trazer problemas na otimização dos modelos.

Na tentativa de obter uma função de divergência que funcione para mais casos, [65] propõe o uso da função de divergência chamada de *Earth-Mover* (ou *Wasserstein-1*). Esta nova função indica o quanto de massa[1] deve ser transportada de um ponto para outro para aproximar a distribuição dos dados de saída e a dos dados reais. A distância obtida é o custo ótimo de transporte de massa. Outra mudança é no valor de saída de D, que antes trabalhava como um classificador e agora passa a trabalhar como um pontuador, dando uma nota de anomalia para cada instância analisada, onde pontuações maiores indicam que a amostra analisada pertence aos dados reais. D agora passa a ser chamado de Crítico (para facilitar o entendimento das fórmulas, ele ainda será referenciado pela letra D).

Para obter a função de valor da WGAN é necessário utilizar a dualidade de *Kantorovich-Rubinstein*, que calculará a distância *Earth-Mover*. Outra condição a ser satisfeita é que D seja uma função *K-Lipschitz*[2], garantindo suavidade nos valores de saída. Satisfeitas as condições, a função de valor da WGAN pode ser descrita por [65]:

$$\min_G \max_{D \in \mathcal{D}} W(D, G) = \mathbb{E}_{x \sim \mathbb{P}_r}[D(x)] - \mathbb{E}_{\tilde{x} \sim \mathbb{P}_G}[D(\tilde{x})], \tag{3.2}$$

onde $\mathcal{D}$ é o conjunto de funções *1-Lipschitz* que satisfazem o problema e $\mathbb{P}_r$ e $\mathbb{P}_G$ seguem as definições já descritas. Os valores obtidos por esta nova função tem uma melhor correlação, em relação à função de custo da *Vanilla* GAN, com a qualidade das amostras produzidas.

Para garantir a restrição de *Lipschitz* em D, a implementação da WGAN utiliza uma constante que limita seus pesos, podendo afetar a aprendizagem de D em alguns casos. A WGAN-GP busca solucionar este problema propondo outra maneira de garantir a restrição de *Lipschitz*, por meio da Penalidade de Gradiente. Este método mais flexível calcula a norma do gradiente de D em pontos interpolados entre amostras reais e geradas, penalizando desvios significativos de um gradiente unitário. A função de valor obtida conta com um novo termo capaz de regularizar o comportamento de D, ela pode ser descrita por [64]:

$$\min_G \max_D L(D, G, \lambda) = \mathbb{E}_{\tilde{x} \sim \mathbb{P}_G}[D(\tilde{x})] - \mathbb{E}_{x \sim \mathbb{P}_r}[D(x)] + \lambda \mathbb{E}_{\hat{x} \sim \mathbb{P}_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2], \tag{3.3}$$

onde λ é o coeficiente que controla a intensidade da penalidade de gradiente e $\mathbb{P}_{\hat{x}}$ é definido implicitamente por amostras uniformes ao longo de linhas retas entre pares de pontos amostrados da distribuição de dados $\mathbb{P}_r$ e da distribuição do gerador $\mathbb{P}_G$.

[1] Utiliza-se o termo massa devido ao nome da função (*Earth-Mover*) que remete à ação de terraplanagem, movendo terra para igualar as distribuições. Neste caso, a massa seriam os valores em cada distribuição.

[2] Uma função é K-*Lipschitz* se ela possui uma taxa de variação limitada, ou seja, a mudança na saída é no máximo K vezes a mudança na entrada. Matematicamente, para uma função f, quaisquer dois pontos x e y do domínio devem satisfazer $|f(x) - f(y)| \leq K|x - y|$, onde K é constante real positiva.

A arquitetura final da WGAN-GP pode ser definida pela Figura 9. O espaço latente amostrado z é transformado, utilizando-se G, em uma amostra sintética $G(z)$. D receberá como entrada amostras sintéticas e amostras reais (x), produzindo uma nota que indica se a amostra analisada é real (valores maiores) ou sintética (valores menores). No contexto da detecção de anomalia, esta nota é chamada de pontuação de anomalia, e espera-se que amostras anômalas produzam pontuações diferentes das normais. Este valor de saída será utilizado nas funções de perda específicas para D e G, atualizando os modelos e contribuindo para suas aprendizagens.

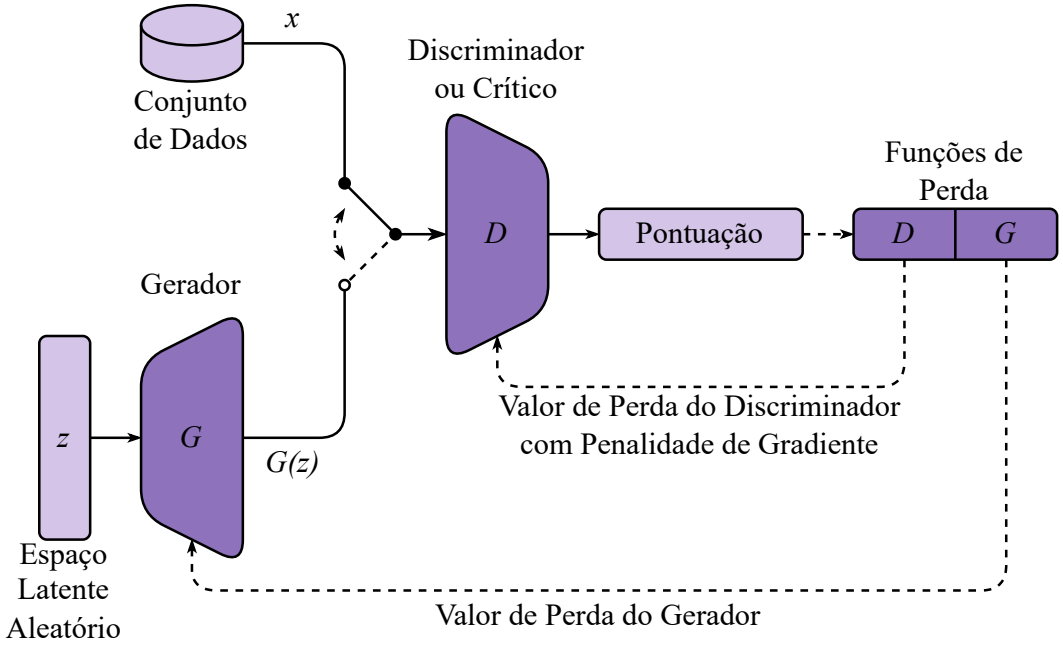

Figura 9 – Arquitetura WGAN-GP. Fonte: elaboração própria.

As mudanças propostas na WGAN-GP promovem uma arquitetura com aprendizagem mais estável, evitando problemas como não convergência e de gradientes desvanecentes. Estas mudanças geram uma arquitetura com a capacidade de modelagem melhor, diminuindo a possibilidade de colapso de modo. Outro benefício apresentado é que a função de perda apresenta correlação com a qualidade das amostras, na qual menores valores de perda implicam em melhores amostras, indicando a diminuição da distância entre as distribuições analisadas [64]. O desempenho fornecido pela arquitetura GAN pode ser aproveitado por diferentes aplicações.

3.3 Aplicações

As GANs têm demonstrado sua versatilidade em diversas aplicações, destacando-se em cenários que envolvem a geração de dados sintéticos, a síntese de imagens e a detecção de anomalias, por exemplo. A capacidade da GAN de aprender distribuições complexas de dados e gerar amostras realistas permite resolver problemas em situações de escassez

de dados. A seguir, serão explorados exemplos dessas aplicações, destacando o impacto da arquitetura GAN em diferentes domínios.

3.3.1 Geração de Dados

No início da pandemia de COVID-19, havia poucas imagens radiográficas de tórax disponíveis para o treinamento de modelos de DL capazes de diagnosticar a doença. Buscando solucionar este problema, Waheed *et al.* [66] propuseram a CovidGAN, um modelo baseado na arquitetura GAN com Classificador Auxiliar (ACGAN). Esta solução gera imagens sintéticas de raios-x de tórax para complementar o conjunto de dados de treinamento. O aumento do conjunto de dados permitiu que o modelo de diagnóstico aumentasse sua Acurácia de 85% para 95% na detecção de COVID-19. Os resultados obtidos indicam que a arquitetura GAN foi capaz de capturar as características relevantes dos dados reais. Este trabalho evidencia como a GAN pode ser usada para superar limitações de conjuntos de dados limitados em tamanho.

3.3.2 Geração de Imagens Detalhadas

A StyleGAN, proposta por Karras *et al.* [67], utiliza conceitos de transferência de estilo para apresentar uma nova arquitetura para geradores de GAN. A abordagem introduz o conceito de espaço latente intermediário, separando entre atributos de maior importância e variações menores. A arquitetura proposta permite controle sobre a síntese de imagens em diferentes escalas, melhorando sua qualidade e facilitando a interpolação entre estilos. Uma das aplicações é a síntese de retratos humanos em alta resolução e com maior controle das características faciais específicas. Estas características destacam a arquitetura como um método capaz de gerar dados sintéticos com detalhamento em diferentes contextos.

3.3.3 Texto para Imagem

Gerar imagens foto-realistas a partir de descrições textuais é uma tarefa desafiadora devido à alta dimensionalidade dos dados e a dificuldade de treinar modelos que consigam mapear e representar esses dados. Zhang *et al.* [68] propuseram a arquitetura StackGAN, que divide a síntese das imagens em dois estágios, abordando o problema de forma hierárquica. O primeiro estágio utiliza um modelo baseado na arquitetura GAN para gerar imagens de baixa resolução que esboçam as formas e cores básicas com base na descrição textual. O segundo estágio utiliza essas imagens de baixa resolução e outro modelo também baseado na arquitetura GAN para refinar as imagens adicionando detalhes foto-realistas, melhorando a qualidade das imagens geradas. Este trabalho reforça a aplicabilidade da arquitetura GAN em contextos onde a geração de dados sintéticos detalhados é essencial.

3.3.4 Detecção de Anomalias em Imagens

A necessidade de detecção de anomalias em imagens médicas motivou o desenvolvimento da f-AnoGAN, um modelo não supervisionado baseado na arquitetura GAN. Proposto por Schlegl *et al.* [36], este modelo busca resolver desafios associados à geração de imagens normais e à detecção de padrões anômalos. Ele utiliza um modelo GAN treinado apenas com dados normais para capturar a variabilidade natural das imagens e treina um codificador que mapeia novas entradas para o espaço latente. Esta abordagem permite reconstruir versões normais de imagens de entrada e detectar anomalias baseando-se nas diferenças residuais entre as imagens originais e reconstruídas. No contexto em que foi aplicada, a f-AnoGAN apresentou maior precisão na detecção de anomalias em relação a métodos alternativos, demonstrando o potencial das variações da arquitetura GAN.

3.3.5 Detecção de Anomalias em Redes de Computadores

A arquitetura GAN também se destacou na detecção de anomalias em redes de computadores devido à sua capacidade de gerar dados sintéticos e aprender distribuições de dados. Lim *et al.* [16] produziram uma revisão que aborda o futuro do uso da GAN na detecção de anomalias, com ênfase em ambientes com escassez de dados. O artigo destaca como modelos baseados na GAN são utilizados para aprender a distribuição normal do tráfego da rede, assim detectando padrões que desviam dessas distribuições (*outliers*). Esta abordagem é útil em cenários reais, onde as amostras normais normalmente estão em maior abundância do que as anômalas. O trabalho também compara diferentes modelos baseados em GAN, propondo melhorias para estes sistemas.

Outra arquitetura que tem se destacado no contexto de NN são os Transformadores. Criada inicialmente para tarefa de traduções, esta arquitetura mudou o modo como redes neurais processam informações sequenciais, utilizando o mecanismo de atenção [1]. Esta arquitetura foi capaz de desempenhar tarefas como as de visão computacional e detecção de anomalias [15].

4 TRANSFORMADOR

Desenvolvido por pesquisadores do Google [1], o Transformador é uma arquitetura inicialmente pensada para tarefa de tradução. Ela é baseada somente em mecanismos de atenção, dispensando completamente o uso de recorrências e convoluções. Estas alterações permitem que ele seja paralelizável, requerendo menos tempo de treinamento em sistemas que possuam essa característica. Nos últimos dois anos a arquitetura ganhou notoriedade do público geral por ser utilizada no modelo de uso geral conhecido como Transformador Generativo Pré-treinado (GPT, do inglês *Generative Pre-trained Transformer*) [69], [70], [71], [72].

A arquitetura do Transformador é resultado da combinação de outros elementos menores. Para compreender sua execução e vantagens, é primeiramente necessário entender a base de funcionamento dele, que é o mecanismo de atenção descrito na Seção 4.1. A Seção 4.2 exibe os demais componentes e suas combinações, que são utilizadas com diferentes propósitos na arquitetura. Alguns exemplos de possíveis aplicações usando Transformadores serão exibidos na Seção 4.3

4.1 Mecanismo de Atenção

O Mecanismo de Atenção é uma parte essencial da arquitetura do Transformador. Este mecanismo permite que o modelo foque em diferentes partes de uma entrada ao processá-la, atribuindo pesos variáveis a diferentes elementos com base em sua relevância. Esta atribuição é feita para capturar as relações e dependências entre os elementos de uma sequência de entrada [14].

Como pode ser visto na Figura 10, este mecanismo funciona processando três componentes principais: consultas, chaves e valores [73]. Cada elemento da entrada é representado por um vetor e o objetivo é determinar a relevância entre as consultas e as chaves para pesar os valores de atenção correspondentes. Para isso, calcula-se um produto escalar entre cada consulta e todas as chaves. Os resultados são escalados pelo tamanho do vetor das chaves e normalizados por uma função *softmax*, que gera um conjunto de pesos representando a importância relativa de cada chave em relação à consulta. A função *softmax* converte um vetor de números em probabilidades, onde estes números podem variar de $-\infty$ a $+\infty$, mas a saída será um vetor de probabilidades, onde o menor valor possível é 0 e a somatória de todos os elementos é 1. Os pesos calculados são então aplicados aos valores correspondentes, produzindo uma saída ponderada. O processo deste cálculo pode ser visualizado na Figura 11 (a), na parte da Atenção de Produto Escalar Escalado.

O processo pode ser comparado a uma busca *hash*. Assim como este tipo de busca encontra uma correspondência exata baseada em uma chave única, o mecanismo analisado computa a similaridade entre a consulta e todas as chaves, mas de forma mais flexível, permitindo considerar correspondências parciais ou graduais [1]. Essa flexibilidade é crítica para capturar relações complexas em dados sequenciais, especialmente quando a relevância depende do contexto ou posição do elemento na entrada.

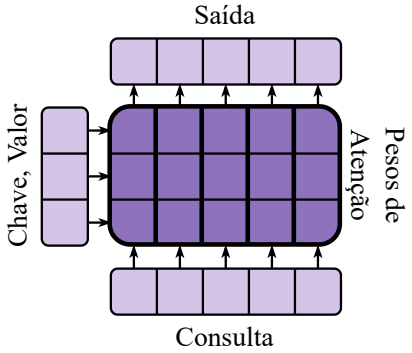

Figura 10 – Camada básica de atenção. Fonte: elaboração própria.

A auto-atenção é uma variante do mecanismo de atenção que permite que cada posição de uma sequência de entrada se relacione com todas as outras posições, inclusive consigo mesma [1]. No caso do Transformador, ela substitui o uso de Redes Neurais Recorrentes, que processam os elementos sequencialmente. Ao fazer isso, elimina-se a dependência sequencial, permitindo que o modelo processe todos os elementos em paralelo. Essa característica não apenas acelera o treinamento, mas também melhora a capacidade do modelo de capturar dependências de longo alcance entre palavras, algo que as Redes Neurais Recorrentes frequentemente encontram dificuldades em representar. A capacidade da arquitetura ser paralelizável é especialmente aproveitada por unidades de processamento gráfico e de tensor, devido às suas arquiteturas feitas especificamente para este tipo de cálculo matemático.

A atenção multicabeça, apresentada na Figura 11 (b), expande o mecanismo de atenção ao dividir as entradas em várias representações independentes, chamadas de cabeças. Cada cabeça aplica uma transformação linear às consultas, chaves e valores, gerando projeções distintas. Em seguida, cada uma dessas projeções passa pelo mecanismo de atenção de produto escalar escalado, calculando as saídas ponderadas para cada cabeça. Essas saídas são concatenadas e passam por uma última camada linear, combinando as informações aprendidas pelas diferentes cabeças. Essa abordagem permite que o modelo aprenda relações em diferentes subespaços de representação, melhorando sua capacidade de capturar padrões complexos.

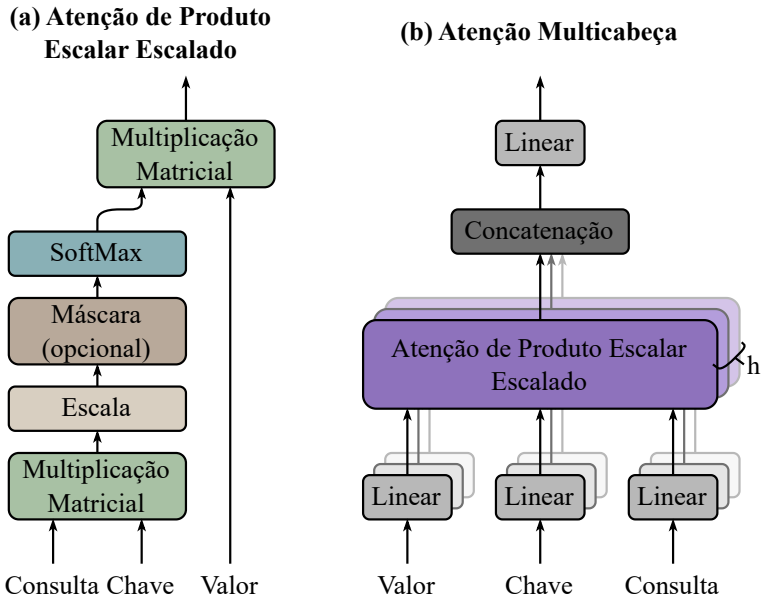

Figura 11 – Atenção de Produto Escalar Escalado (a) e Atenção Multicabeça (b). Fonte: adaptado de Vaswani *et al.* [1].

4.2 Arquitetura

A Figura 12 apresenta os componentes da arquitetura do Transformador, contando com um codificador e um decodificador. Cada um desses dois componentes possui elementos internos, sendo [1]:

- **Incorporação e Codificação Posicional**: a entrada do Transformador é inicialmente convertida em vetores densos por meio de uma camada de incorporação, que associa cada elemento de entrada a uma representação vetorial contínua. Para que o modelo compreenda a ordem dos elementos na sequência, vetores de codificação posicional, que utilizam funções seno e cosseno para gerar valores determinísticos baseados na posição de cada elemento, são somados às incorporações. Essa combinação permite que o Transformador processe informações sequenciais, mesmo sem o uso de mecanismos recorrentes.

- **Adição e Normalização**: a arquitetura do Transformador utiliza conexões residuais para melhorar o fluxo de informações e estabilizar o treinamento. Em cada subcamada, os resultados do processamento são somados à entrada original (adição residual) e o resultado passa por uma normalização de camada. Esse processo reduz o risco de gradientes desvanecentes ou explosivos, melhorando a eficiência da aprendizagem e a estabilidade do modelo.

- **Camada Base de Atenção**: esta camada base combina a atenção multicabeça com o mecanismo de adição e normalização. As consultas, chaves e valores passam pelas

diferentes cabeças de atenção, gerando saídas ponderadas que são concatenadas e processadas por uma camada linear. O resultado é somado à entrada original e normalizado, formando uma estrutura robusta para capturar dependências entre os elementos da sequência.

- **Atenção Cruzada**: presente no decodificador, as consultas vêm do decodificador, enquanto as chaves e valores são provenientes do codificador. Isso permite que cada posição no decodificador "atenda"a todas as posições da sequência codificada, criando uma ponte entre a entrada e a saída. Este mecanismo é essencial em tarefas como tradução, onde a saída depende diretamente da sequência de entrada.

- **Auto-atenção Global**: usada no codificador e permite que cada posição de uma sequência se relacione com todas as outras posições da mesma sequência. Isso é crucial para capturar dependências de longo alcance. Nesse contexto, todas as consultas, chaves e valores são gerados a partir da mesma camada, possibilitando que o modelo compreenda relações entre palavras independentemente da distância.

- **Auto-atenção Causal**: no decodificador, este componente garante que cada posição na sequência só possa acessar posições anteriores ou atuais. Isso é implementado por meio de uma máscara que impede que posições futuras influenciem nos cálculos dos valores de atenção. Esse mecanismo preserva a natureza autorregressiva da geração de sequência, fundamental para tarefas como predição de texto.

- **Alimentação Direta**: após as operações de atenção, cada posição da sequência passa por uma rede de alimentação direta (*feed forward*), composta por duas camadas lineares separadas por uma função de ativação ReLU[1]. A saída é somada à entrada original e normalizada, garantindo que o modelo aprenda representações mais complexas para cada posição da sequência de forma independente.

Compostos por camadas que combinam atenção e redes de alimentação direta, o Transformador organiza sua arquitetura em duas partes principais: o codificador e o decodificador. Cada um deles é formado por N camadas empilhadas, permitindo maior capacidade de aprendizagem e representatividade. Essa estrutura modular e hierárquica permite ao Transformador lidar com tarefas complexas, como tradução e geração de texto. Segue uma descrição mais detalhada de cada componente [1]:

- **Codificador**: o codificador do Transformador também é composto por N camadas idênticas empilhadas, cada uma contendo duas subcamadas principais: auto-atenção global e uma rede de alimentação direta. A entrada da sequência, após passar pela incorporação e codificação posicional, é processada pelo codificador, que

[1] A Unidade Linear Retificada (ReLU, do inglês *Rectified Linear Unit*) é uma função de ativação que tem como saída o valor de entrada caso ele seja positivo, e zero caso contrário.

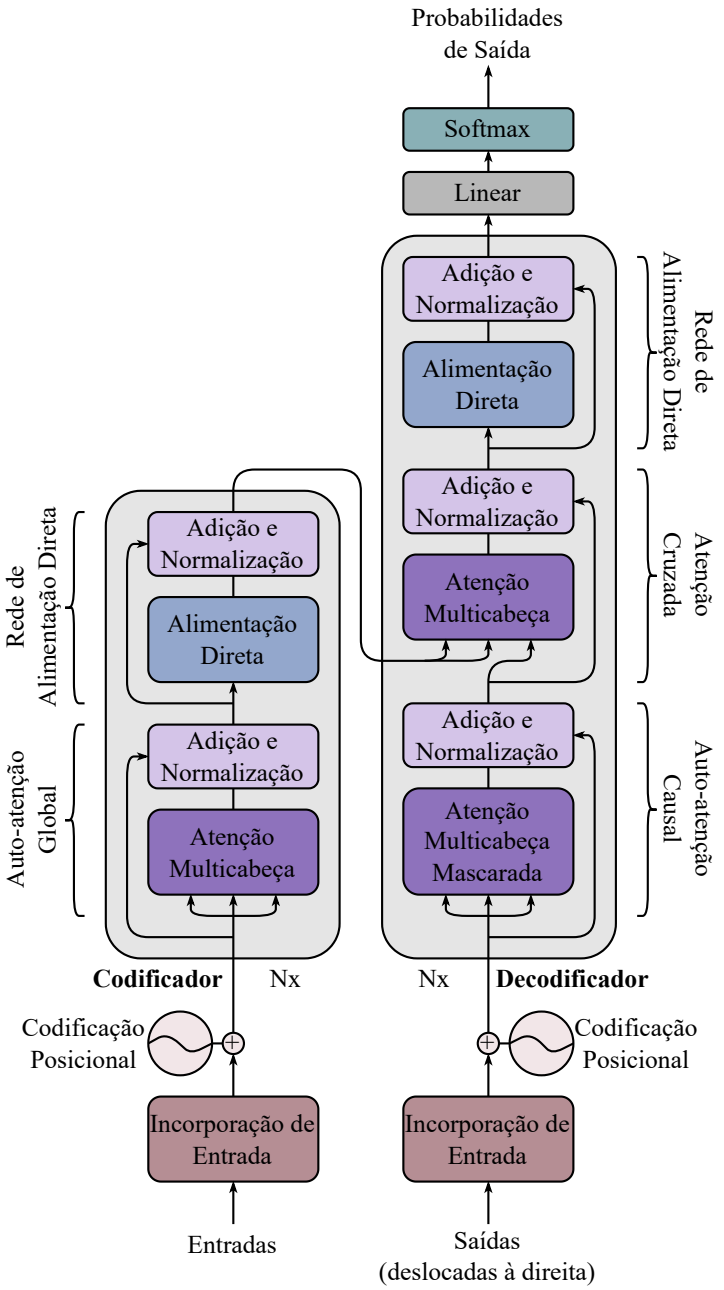

Figura 12 – Arquitetura do Transformador. Fonte: adaptado de Vaswani *et al.* [1].

aplica a auto-atenção global para capturar dependências entre todos os elementos da sequência. Após a atenção, a rede de alimentação direta refina as representações em cada posição da sequência. Em cada subcamada, são utilizadas conexões residuais e normalização de camada para estabilizar o treinamento e melhorar a eficiência do modelo. O resultado final do codificador é uma representação rica e contextualizada da sequência de entrada, que é passada para o decodificador.

- **Decodificador**: o decodificador também é formado por N camadas empilhadas, porém com três subcamadas principais em cada uma delas: auto-atenção causal, atenção cruzada e uma rede de alimentação direta. A auto-atenção causal garante

que a geração da saída seja autorregressiva, respeitando a ordem sequencial. A atenção cruzada conecta o decodificador ao codificador, permitindo que cada posição da sequência de saída utilize as informações aprendidas do codificador. A rede de alimentação direta refina as representações da saída em cada posição. Assim como no codificador, as conexões residuais e a normalização de camada são aplicadas em cada subcamada, garantindo robustez e estabilidade ao treinamento. A saída do decodificador é então processada por uma camada linear, transformando os vetores de saída em *logits*, que são os valores brutos gerados pela camada. Estes valores serão passados por uma função *softmax*, sendo convertidos em probabilidades para a predição do próximo elemento da sequência. O decodificador gera a sequência de saída final, incorporando as informações contextuais aprendidas ao longo do processo.

Apesar de ter sido desenvolvido para a tarefa de tradução, o Transformador também foi capaz de desempenhar em outras aplicações [15]. Isso se dá devido à robustez da arquitetura, que possibilita que os modelos aprendam os padrões dos dados, mesmo em conjuntos complexos.

4.3 Aplicações

A arquitetura do Transformador ofereceu avanços em tarefas que envolvem a modelagem de dependências em sequências de dados. Esta capacidade de modelar dependências possibilitou que a arquitetura fosse utilizada em problemas de tradução automática, geração de texto e detecção de anomalias, por exemplo. A seguir, são exploradas algumas das aplicações desta arquitetura, exibindo exemplos que demonstram sua eficácia e impacto em diferentes domínios.

4.3.1 Tradução

Um dos principais avanços demonstrados pela arquitetura do Transformador, promovido por Vaswani *et al.* [1], foi demonstrado na tarefa de tradução automática, onde o modelo foi treinado para traduzir entre diversos pares de línguas, incluindo inglês e francês. Utilizando um mecanismo de atenção escalada, o Transformador alcançou um novo patamar de qualidade ao superar modelos baseados em redes recorrentes e convolucionais. Em experimentos realizados, o modelo obteve uma pontuação BLEU[2] de 41,8 na tradução do inglês para o francês, estabelecendo um novo estado da arte com um custo computacional menor que seus predecessores. Esses resultados destacam a eficiência e precisão

[2] O Substituto de Avaliação Bilíngue (BLEU, do inglês *Bilingual Evaluation Understudy*), é uma métrica para avaliar automaticamente textos produzidos por máquina. A pontuação é produzida ao medir-se a similaridade do texto traduzido por máquina com um conjunto de traduções de referência.

da arquitetura em capturar dependências complexas entre sequências de segmentos de palavras de entrada e saída.

4.3.2 Transformadores Generativos Pré-treinados

Os modelos GPT, desde sua introdução, revolucionaram a forma como AI é aplicada a tarefas de processamento de linguagem natural. O GPT-1 [69] demonstrou o poder do pré-treinamento em larga escala seguido de ajustes finos, estabelecendo uma abordagem para transferência do aprendizado. O GPT-2 [70] ampliou a capacidade do modelo, introduzindo 1,5 bilhão de parâmetros e sendo capaz de operar em configurações de *zero-shot*, onde o modelo executa tarefas sem treinamento supervisionado prévio específico. Com o GPT-3 [71], contendo 175 bilhões de parâmetros, foram obtidos avanços na geração de texto, resumo, tradução e programação, consolidando-se como uma ferramenta versátil de uso geral. O GPT-4 [72], além de aprimorar o desempenho em testes de referência linguísticos, integrou processamento multimodal, permitindo entrada de texto e imagens, e atingiu níveis comparáveis ao desempenho humano em avaliações profissionais e acadêmicas. Todos esses modelos utilizam a arquitetura de Transformadores como base, explorando sua capacidade de auto-atenção para capturar relações contextuais entre segmentos de texto em sequências longas de texto, possibilitando seu desempenho superior em uma ampla gama de tarefas.

4.3.3 Detecção de Anomalias em Séries Temporais

O modelo de Transformador Temporal Variável (VTT, do inglês *Variable Temporal Transformer*), proposto por Kang *et al.* [17], representa um avanço na detecção de anomalias em séries temporais multivariadas. Desenvolvido para superar limitações de arquitetura tradicionais, como redes recorrentes e convolucionais, o VTT emprega um mecanismo de auto-atenção inovador que integra dependências temporais e correlações entre variáveis. Aplicado em diversos conjuntos de dados reais e sintéticos, o modelo demonstrou alta precisão na identificação de anomalias ao reconstruir séries temporais e medir desvios em relação aos dados originais. A eficácia do VTT é reforçada pela introdução de um módulo interpretativo, que permite rastrear as causas das anomalias com base nas mudanças nos mapas de atenção, oferecendo percepções valiosas para setores como manufatura, tecnologia da informação e monitoramento industrial.

Apesar do seu desempenho, a arquitetura do Transformador, assim como outros modelos complexos de DL, apresentam desafios relacionados à interpretabilidade e explicabilidade, devido às suas naturezas altamente intrincadas e às milhões de iterações internas que os tornam poderosos, mas opacos.

5 INTELIGÊNCIA ARTIFICIAL EXPLICÁVEL

O uso de AI está cada vez mais integrado nas atividades diárias da sociedade, possibilitando a tomada de decisões baseada em dados, sendo muitas vezes confidenciais ou sensíveis [19]. A crescente aplicação desses modelos acaba exigindo que seu funcionamento seja explicado, requisito nem sempre possível em modelos mais complexos, como os que usam NN [74]. Pensando em auxiliar nesse problema, as técnicas de Inteligência Artificial Explicável (XAI, do inglês *Explainable Artificial Intelligence*) surgem como alternativa para compreender o funcionamento desses modelos, auxiliando seus desenvolvimentos e não interferindo em seus desempenhos [75].

5.1 Interpretabilidade e Explicabilidade

Duas nomenclaturas comumente usadas em trabalhos que abordam XAI são explicabilidade e interpretabilidade. Apesar de semelhantes, esses termos têm diferentes significados e usos, já que métodos interpretáveis são explicáveis se humanos conseguem compreender suas operações. Segue a definição dos termos relacionados [18]:

- **Explicabilidade:** relacionada com o processo de elucidar os mecanismos de tomada de decisão do modelo, possivelmente exibindo como as entradas e saídas são matematicamente interligadas. Fornece a habilidade de entender o **porquê** da tomada de decisão, possibilitando a interpretação e descrição do funcionamento interno do modelo. Uma técnica explicável busca sumarizar as razões para uma decisão do modelo.
- **Interpretabilidade:** revela as propriedades intrínsecas do modelo, possibilitando a compreensão de **como** ele toma a decisão. Nas técnicas interpretáveis, o foco é nos mecanismos internos do modelo e qual o funcionamento deles para tomada de decisão.

Outra forma de dividir as definições é em relação ao seu objetivo em relação ao usuário [76]. A explicabilidade fornece informações para um público-alvo para atender uma necessidade, enquanto a interpretabilidade é o grau em que as informações fornecidas fazem sentido para o conhecimento do domínio do público-alvo em questão.

5.2 Objetivos

Modelos de AI podem ser divididos em caixa preta, cinza e branca. Modelos caixa branca/cinza possuem alta explicabilidade e interpretabilidade, mas, de modo geral, me-

nor desempenho, como modelos de Regressão Linear, K-vizinhos mais Próximos e Árvores de Decisão [51]. Modelos caixa preta, pelo contrário, apresentam, geralmente, alto desempenho, mas baixa explicabilidade e interpretabilidade, como DNNs e Transformadores [77]. Por possuírem, na maioria dos casos, melhor desempenho, modelos caixa preta são escolhidos para implementações com maior frequência, muitas vezes em aplicações que interagem diretamente com humanos ou dados sensíveis [78]. Cria-se, então, uma preocupação da comunidade científica em desenvolver métodos que auxiliem na compreensão desses modelos, a fim de torná-los mais seguros.

O objetivo geral das técnicas de XAI é desmistificar sistemas de AI complexos, particularmente os aplicados em ambientes críticos, como de saúde, militar, bancário e de segurança [79]. Isso abre uma janela em modelos caixa preta, tornando o processo de decisão do modelo humanamente interpretável tanto para especialistas no domínio, quanto para usuários finais. Existem diversas perspectivas de pesquisa que destacam os objetivos da XAI, como capacitar indivíduos contra potenciais riscos em decisões automatizadas, melhorar a toma de decisões baseada em informações, proteger contra vulnerabilidade de segurança e alinhar processos algorítmicos com valores humanos [80]. Métodos XAI buscam diminuir a lacuna entre o avanço de técnicas computacionais e a compreensão humana, para isso, diferentes artigos separam as diferentes perspectivas do uso da XAI.

Saeed *et al.* [76] realizaram uma revisão que aborda os desafios da XAI e as futuras direções de pesquisa. Contribuíram também categorizando cinco perspectivas do uso das técnicas de XAI, sendo: a Regulatória que destaca a conformidade legal do modelo, respeitando, por exemplo, a Lei Geral de Proteção de Dados Pessoais [81]; a Científica que aborda como esses modelos podem ser utilizados para produzir conhecimento; a Industrial que trata da regulamentação do uso desses modelos em empresas; a do Desenvolvimento para auxiliar desenvolvedores no aprimoramento dos seus modelos; e a do Usuário Final e Social que enfatiza a necessidade de explicações que tornem a decisão dos modelos mais humanamente compreensíveis.

Capuano *et al.* [82] apresentam uma análise sistemática sobre aplicações XAI no domínio da cibersegurança, abordando áreas como detecção de intrusões e outras ameaças. Os autores se baseiam na divisão de princípios da XAI proposta pelo Instituto Nacional de Padrões e Tecnologia [83]. Os quatro princípios fundamentais são: Explicação, que trata sobre os sistemas fornecerem evidências ou razões para os resultados que produzem; Significância, que diz sobre os sistemas fornecerem explicações compreensíveis para os usuários; Precisão da Explicação, que enfatiza a importância das explicações refletirem corretamente os processos do sistema para geração das saídas; e o de Limites do Conhecimento, que ressalta que os sistemas só devem operar nas condições em que foram projetados.

Moustafa *et al.* [84] realizaram uma revisão abrangente sobre o uso de XAI em

sistemas de detecção de intrusão para rede de Internet das Coisas. Os autores discutem como os métodos XAI podem aumentar a confiança e a interpretabilidade em modelos de AI, destacando as oportunidades da área e soluções já existentes. Eles também separam os usuários que se beneficiam da XAI em cinco grupos, sendo: Agências Regulatórias, que certificam a conformidade do modelo com a regulamentação vigente; Especialistas no Domínio, que abordam como os modelos podem gerar conhecimento, aumentado sua confiabilidade; Gerentes, garantindo a conformidade do modelo com os regulamento da empresa; Desenvolvedores e Cientista de Dados, garantindo melhorias na eficiência do modelo, pesquisando recursos e possíveis novas funcionalidades; Pessoas Afetadas pelas Decisões do Modelo, que devem estar cientes das decisões do modelo e como elas ocorrem.

Neste trabalho serão utilizados métodos XAI com o objetivo de melhorar a compreensão e confiabilidade de modelos de ML aplicados na detecção de anomalias em redes de computadores. Baseando-se nas definições e perspectivas apresentadas neste capítulo, essas técnicas serão empregadas para desmistificar os processos de decisão de modelos complexos, promovendo maior explicabilidade. O foco será nas perspectivas Científica e de Desenvolvimento, oferecendo explicações compreensíveis de como o modelo toma as decisões, auxiliando no processo de ajuste do modelo e entendendo como ele relaciona os dados para produzir conhecimento.

5.3 Métodos

Diversos métodos de XAI foram desenvolvidos ao longo dos anos, cada um com objetivos distintos, mas sempre focados em aumentar a compreensão dos modelos de ML. Esses métodos podem ser classificados em três categorias principais, com base em seus objetivos específicos [18]:

- **Explicabilidade de Dados:** busca compreender como os dados utilizados no treinamento influenciam o comportamento do modelo, analisando aspectos como importância das características e possíveis vieses presentes no conjunto de dados. Alguns exemplos de métodos são análise exploratória de dados, engenharia de características e padronização da descrição de conjunto de dados.

- **Explicabilidade de Modelo:** se concentra em tornar o próprio modelo mais transparente, utilizando arquiteturas interpretáveis ou simplificadas que permitam uma análise direta de seu funcionamento. Dentre os métodos deste tipo pode-se citar a escolha de modelos inerentemente interpretáveis, unir a predição do modelo com a explicação e a explicabilidade por meio de ajustes de arquitetura.

- **Explicabilidade *Post-hoc*:** fornece explicações após o treinamento do modelo, utilizando técnicas como visualizações, métodos baseados em perturbação ou apro-

ximações locais para elucidar os fatores que levam a uma decisão específica. Como exemplos deste tipo de método pode-se citar os de atribuição, visualização e baseado na teoria dos jogos.

Cada uma dessas abordagens atende a diferentes necessidades, condições e públicos. Neste trabalho, métodos que objetivam a explicabilidade de dados serão usados indiretamente na escolha do conjunto de dados e das características dele que serão utilizadas. Métodos de explicabilidade do modelo não serão utilizados, pois os modelos deste trabalho foram escolhidos com base no desempenho e não em sua explicabilidade. O foco estará nos métodos *post-hoc*, auxiliando no entendimento de como os dados se relacionam dentro do modelo para produzir as saídas. Os três métodos *post-hoc* escolhidos serão explicados nas próximas subseções.

5.3.1 Explicações Aditivas de Shapley

O método de Explicações Aditivas de Shapley (SHAP, do inglês *Shapley Additive Explanations*), proposto por Lundberg *et al.* [85], é baseado na Teoria dos Jogos. Este método busca identificar a contribuição de cada característica na predição de um modelo de ML. Inspirado pelos valores de Shapley [86], o SHAP analisa todas as possíveis combinações de características, considerando o impacto de uma característica específica na predição quando ela é adicionada ou removida das diferentes combinações. O método representa as predições como uma soma aditiva de valores atribuídos a cada característica. Esses valores indicam como cada característica contribui positiva ou negativamente para o resultado do modelo. Isso permite uma análise detalhada e interpretável das decisões tomadas, inclusive em modelos complexos como NN.

A Figura 13 (a) exibe uma das possíveis saídas SHAP que busca mostrar a influência dos valores das características em relação aos valores SHAP. O eixo x representa os valores SHAP, o eixo y todas as características. Cada ponto no gráfico representa um valor SHAP para uma predição em relação a uma característica. Pontos mais à direita influenciaram positivamente a predição do modelo, enquanto pontos mais à esquerda, negativamente. Existe também uma barra vertical com um gradiente de cores que indica os valores de cada característica em cada ponto. Neste exemplo, valores altos e médios da Característica 1 influenciaram positivamente na predição, enquanto os baixos não interferiram; valores altos e baixos da Característica 2 influenciaram negativamente, enquanto os médios não interferiram; os valores da Característica 3 interferiram pouco na predição. A saída exibida na Figura 13 (b) indica quanto cada característica influencia na predição, independente de ser positiva ou negativamente. Neste exemplo, a que mais influência é a Característica 1, seguida da 2 e 3.

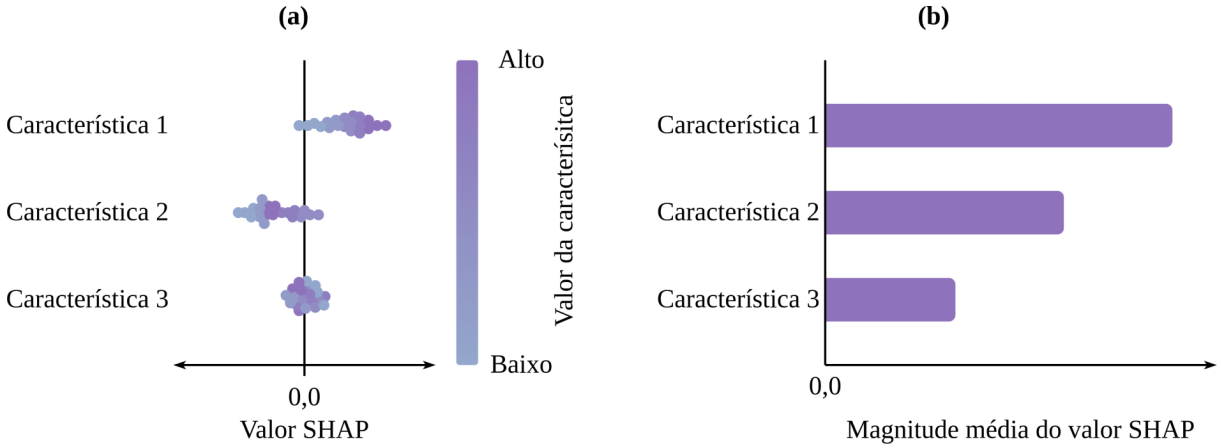

Figura 13 – Exemplos de saída SHAP. Fonte: elaboração própria.

5.3.2 Efeitos Locais Acumulados

O método de Efeitos Locais Acumulados (ALE, do inglês *Accumulated Local Effects*), proposto por Apley *et al.* [87], mede o efeito acumulado de uma característica sobre a predição, independentemente das demais características. O ALE considera a dependência entre as características, fornecendo valores mais confiáveis em casos onde há correlação entre elas. Os resultados são obtidos analisando variações locais no modelo ao alterar uma característica específica em pequenos intervalos, enquanto as demais são mantidas fixas. Essas variações são acumuladas ao longo dos intervalos, resultando em uma curva que representa o impacto médio de característica na predição.

A Figura 14 mostra um dos tipos de saída baseado no método ALE. Complementando o gráfico da Figura 13 (a), essa saída mostra a influência de cada característica na predição do modelo em relação ao seu valor. Cada característica terá seu próprio gráfico, onde o eixo x representa seus valores e o eixo y o valor ALE. Cada ponto no gráfico representa o nível de influência na predição em relação ao valor da característica. Neste exemplo, existe uma maior concentração de valores altos e médios para Característica 1, sendo esses os que mais influenciam positivamente o modelo; já para a Característica 2, há uma maior concentração de valores mais baixos, que influenciam negativamente o modelo; e para a Característica 3 os valores estão bem distribuídos e influenciam pouco o modelo.

5.3.3 Explicações de Modelo Local Interpretável

O método de Explicações de Modelo Local Interpretável (LIME, do inglês *Locally Interpretable Model-Agnostic Explainer*) foi proposto por Ribeiro *et al.* [88] e produz aproximações locais e simplificadas do comportamento do modelo em torno de uma predição específica. Para obter este resultado, o LIME gera perturbações nos dados de entrada e observa como o modelo responde a elas, explicitando a relação entre as características e

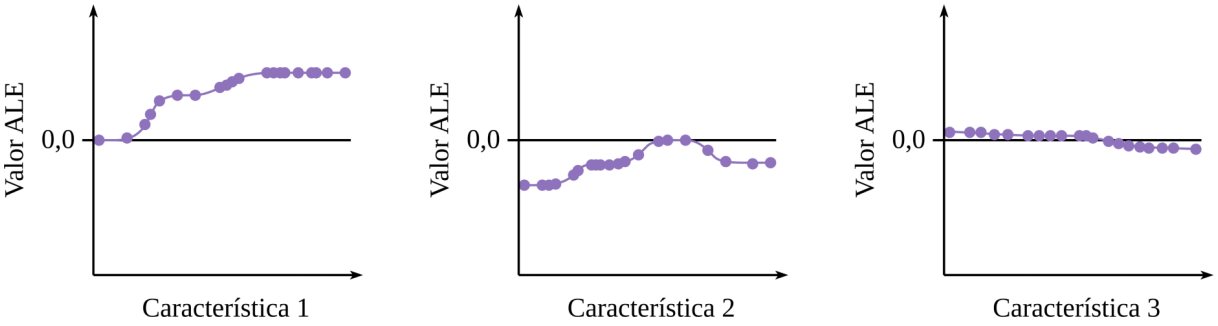

Figura 14 – Exemplo de saída ALE. Fonte: elaboração própria.

a predição de uma vizinhança local do ponto analisado. Este método permite identificar quais características influenciam mais a decisão do modelo em cada instância individual analisada.

A Figura 15 mostra um dos tipos de saída, composta por três partes, baseada no método LIME. A parte (a) exibe as probabilidades de predição por classe, indicando o nível de certeza do modelo para a instância analisada. A parte (b) mostra a pontuação de importância de cada característica na predição em relação às classes, assim como os intervalos de perturbação dos valores das características. A pontuação produzida não está limitada a nenhum intervalo numérico específico, ela somente reflete o efeito estimado que aquela característica tem, especificamente, sobre a predição da instância analisada. A parte (c) mostra os valores originais de cada característica da amostra analisada. Neste exemplo, as probabilidades de predição indicam que o modelo inferiu que a instância analisada era da Classe B, com 75% de certeza. As pontuações de importância indicam que as Características 1 e 3 influenciaram para a predição da Classe B com pontuações de 0,28 e 0,04; respectivamente. A Característica 2 influenciou para a predição da Classe A com uma pontuação de 0,16. Sobre os intervalos de perturbação, a Característica 1 foi perturbada entre 0,74 (valor original) e 1,00; a Característica 2 entre 0,20 e 0,96 (valor original); e a Característica 3 entre 0,28 e 0,69 (valor original). Os valores das características foram 0,74, 0,96 e 0,69 para as Características 1, 2 e 3, respectivamente.

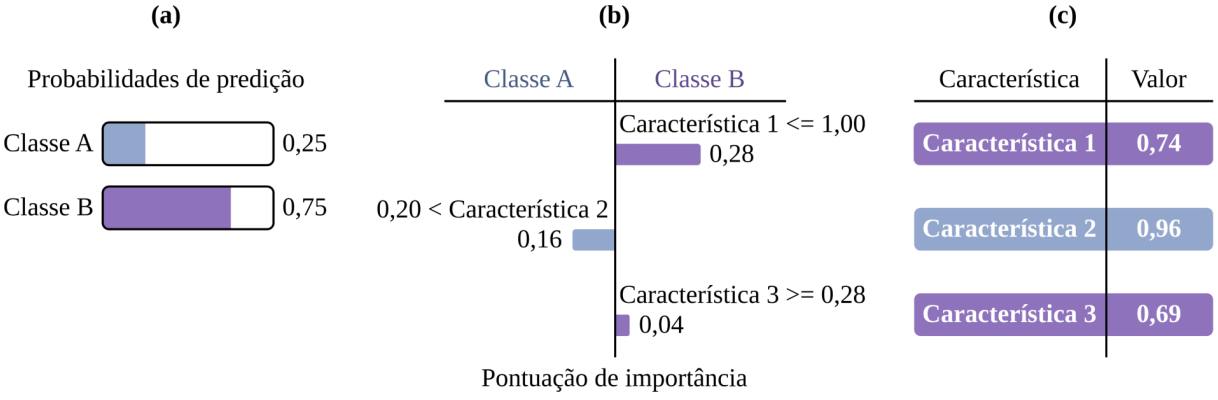

Figura 15 – Exemplo de saída LIME. Fonte: elaboração própria.

6 TRABALHOS RELACIONADOS

A detecção de anomalias, especificamente no tráfego de redes de computadores, tem sido estudada por cientistas nos últimos anos [2], [4], [89], [60]. Duas arquiteturas que se destacaram recentemente foram a GAN e o Transformador, consequentemente sendo alvo de experimentos realizados por pesquisadores da área. Ambas arquiteturas apresentaram resultados promissores, que com a ajuda de métodos XAI, podem ser melhor explicados.

Silva Ruffo *et al.* (2025) [7] propuseram uma metodologia supervisionada baseada em GAN para detecção de intrusões e anomalias em redes baseadas em fluxo IP, com foco em ambientes SDN. A solução implementa uma *Vanilla* GAN, treinada exclusivamente com amostras de tráfego benigno, e utiliza três modelos de aprendizado profundo (Longa Memória de Curto Prazo, Redes Neurais Convolucionais e Redes de Convolução Temporal) na estrutura interna da GAN, explorando suas capacidades individuais para modelagem dos dados do tráfego. O sistema é composto por módulos de coleta, pré-processamento, detecção de anomalias e mitigação, este último automatizado para bloquear fluxos suspeitos e garantir a continuidade dos serviços da rede. Foram conduzidos testes utilizando conjuntos de dados públicos, como CICDDoS2019 [90], CIC-IDS2017 [6] e Orion [91], demonstrando que a abordagem proposta, especialmente as baseadas em Longa Memória de Curto Prazo e Redes Neurais Convolucionais, mitiga o problema de colapso de modelo e mantém um desempenho competitivo com métricas como MCC e F1-*score*. A principal contribuição deste trabalho está na combinação da capacidade de modelagem de padrões das GANs com diferentes modelos internos.

Lent *et al.* [9] propuseram um sistema não supervisionado para detecção de ataques *DDoS* em redes SDN, utilizando GAN com Unidades Recorrentes Fechadas. O modelo utiliza aprendizagem não supervisionada para identificar ataques desconhecidos analisando o tráfego da rede em intervalos de um segundo. Durante o treinamento, o Gerador da GAN aprende a simular o comportamento legítimo do tráfego, enquanto o Discriminador é ajustado para detectar anomalias que diferem do padrão esperado do tráfego. O estudo avaliou o desempenho do sistema com dois conjuntos de dados: Orion [91] e CICDDoS2019 [90]. Os resultados indicaram um F1-*score* de 99% e 98% no primeiro e segundo conjunto de dados, respectivamente. O algoritmo de mitigação alcançou a eliminação de 99% dos fluxos maliciosos. A principal contribuição do trabalho reside no uso de Unidades Recorrentes Fechadas na arquitetura GAN, conseguindo modelar os padrões normais de tráfego.

Yao *et al.* [25] desenvolveram um método escalável e não supervisionado de detecção de intrusões baseado em GAN Bidirecional (BiGAN), voltado para rede da Internet

das Coisas. O modelo foi treinado utilizando dados normais, explorando a distância de Wasserstein para estabilizar o treinamento e melhorar na reconstrução de padrões normais. O método conseguiu minimizar a taxa de falsos positivos utilizando um modelo auxiliar para aprender representações latentes. Avaliado em dois conjuntos de dados (UNSW-NB15 [92] e CIC-IDS2017 [6]), o modelo demonstrou ganhos de 4% em Precisão e redução de alarmes falsos em comparação a outros métodos, chegando 81,9& de Precisão. Este trabalho contribui ao modificar a arquitetura GAN para melhorar a eficiência do sistema de detecção implementado.

Li *et al.* [59] apresentaram o MAD-GAN, uma abordagem baseada em GAN para detecção de anomalias em séries temporais multivariadas. O modelo utiliza redes de Longa Memória de Curto Prazo tanto no Gerador quanto no Discriminador, permitindo capturar dependências temporais nos dados. Uma característica inovadora do MAD-GAN é a utilização conjunta de pontuações de reconstrução e discriminação, integradas em um novo índica, o DR-*score*, que melhora a precisão na identificação de anomalias. O método foi testado em conjuntos de dados reais de sistemas de tratamento e distribuição de água, onde demonstrou superioridade em relação a métodos não supervisionados tradicionais, como Análise de Componentes Principais, Codificador Automático (*Autoencoder*) e outros. Este trabalho destaca-se pela capacidade de modelar interações multivariadas e temporalidades complexas.

Sun *et al.* [55] propuseram o MTS-DVGAN, uma arquitetura generativa adversarial variacional dupla, projetada para detecção de anomalias a partir de séries temporais multivariadas. O modelo utiliza redes de Longa Memória de Curto Prazo no Codificador, Gerador e Discriminador, permitindo capturar dependências temporais e espaciais nos dados. Um diferencial é o uso de um módulo contrastivo, que impõe restrições no espaço latente para melhorar a separação entre dados normais e anômalos, e a introdução de uma nova função de perda para aprimorar a estabilidade do treinamento e minimizar o colapso de modelo. O MTS-DVGAN foi avaliado com três conjuntos de dados, superando outros modelos, como MAD-GAN, em métricas como Precisão e F1-*score*. A abordagem destacou-se pela capacidade de identificação de anomalias próximas à distribuição normal e a robustez frente à instabilidade de gradientes durante o treinamento.

Fu *et al.* [47] introduziram o GANAD, uma abordagem baseada em GAN voltada para detecção de anomalias em redes. O modelo utiliza uma arquitetura baseada em Autoencoder e introduz a normalização espectral e uma penalidade de gradiente para estabilizar o treinamento adversarial, promovendo melhorias no desempenho. O GANAD se destaca por empregar uma estratégia de treinamento que melhora a aprendizagem de padrões anômalos minoritários em meio a dados normais, além de otimizar a computação das perdas de reconstrução e discriminação. Experimentos em conjuntos de dados de tráfego real demonstraram que o modelo supera outros métodos, como MAD-GAN,

em precisão e eficiência, mantendo estabilidade mesmo em cenários de alta dimensionalidade. Este trabalho contribui significativamente ao introduzir técnicas de treinamento aprimoradas e novos componentes na arquitetura GAN aplicada à detecção de anomalias.

Bashar e Nayak [52] apresentaram o TAnoGAN, um modelo baseado em GAN para detecção de anomalias em séries temporais com conjuntos de dados reduzidos. O TAnoGAN utiliza redes de Longa Memória de Curto Prazo tanto no Gerador quanto no Discriminador, permitindo capturar dependências temporais complexas. O processo de detecção baseia-se na reconstrução de dados reais a partir de um espaço latende, com a pontuação de anomalia sendo derivada das perdas de reconstrução e discriminação. Avaliações realizadas com diferentes conjuntos de dados demonstram que o TAnoGAN supera modelos tradicionais e redes neurais, incluindo *Autoencoder* e redes de Longa Memória de Curto Prazo, em métricas como F1-*score* e AUROC. Este trabalho destaca-se por abordar as limitações de modelos que requerem grandes volumes de dados.

Jiang *et al.* [93] introduziram o TransGAN, arquitetura GAN composta exclusivamente por Transformadores, eliminando o uso de convoluções ao trabalhar com imagens. O modelo emprega um Gerador baseado em Transformadores, projetado para aumentar progressivamente a resolução das características, e um Discriminador multi-escala que equilibra a captura de contextos semânticos globais e detalhes de texturas locais. Foram utilizados conjunto de dados para avaliação da geração de imagens, o TransGAN obteve resultados competitivos em métricas como Pontuação de Incepção e Distância de Incepção de Frechet, mostrando-se eficaz em tarefas envolvendo diferentes resoluções de imagens. Este trabalho destaca-se por explorar o potencial dos Transformadores em conjunto com a GAN, testando uma alternativa para o uso de convoluções.

Li *et al.* [94] introduziram o DCT-GAN, uma arquitetura para detecção de anomalias em séries temporais que combina Redes Convolucionais Dilatadas e Transformadores em um modelo generativo adversarial. O modelo utiliza múltiplos Geradores e um único Discriminador para lidar com o problema de colapso de modelo e melhorar a generalização, enquanto os geradores extraem informações em escalas variadas usando blocos de convolução dilatada e módulos baseados em Transformadores. Uma técnica de ponderação dinâmica é empregada para integrar os geradores, permitindo ao modelo lidar com diferentes tipos de anomalias. Avaliações realizadas em conjunto de dados mostraram que o DCT-GAN supera abordagens de ponta, como TAnoGAN e MAD-GAN, em métricas como F1-*score* e AUROC. Este trabalho se destaca pela combinação de múltiplos Geradores com Transformadores para mitigar o colapso de modo.

Gummadi *et al.* [95] propuseram o XAI-IoT, uma estrutura de XAI para aprimorar a detecção de anomalias em sistemas de Internet das Coisas. O modelo combina técnicas de ML, incluindo métodos baseados em Árvores de Decisão e DNN, com sete abordagens de explicabilidade, como SHAP e LIME, para realizar análises globais e locais de importância

de atributos. Avaliações foram realizadas em dois conjuntos de dados reais, onde o modelo proposto não só atingiu taxas de detecção de anomalias satisfatórias em relação a outros modelos, mas também identificou características que diferenciam dados normais de falhas ou ataques, permitindo maior transparência e confiabilidade. Este trabalho se destaca ao oferecer uma ferramenta integrada que combina precisão na detecção com informações interpretáveis, contribuindo para segurança e eficiência do sistema.

Os trabalhos relacionados apresentados demonstram a evolução contínua e as diferentes variações das técnicas de detecção de anomalias. Abordagens como a BiGAN [25] destacam-se por propor uma alteração estrutural da arquitetura GAN, acrescentando um componente codificador auxiliar, enquanto modelos como o TransGAN [93] e DCT-GAN [94] avançam ao incorporar Transformadores para melhorar a generalização das NN. No entanto, nenhum dos métodos revisados explorou de forma integrada o potencial dos Transformadores aliados à arquitetura WGAN-GP, que busca oferecer maior estabilidade no treinamento. O presente trabalho testa a combinação destas arquiteturas, com o objetivo de descobrir se seus potenciais isolados podem ser combinados.

7 ESTUDO DE CASO

A seguir, será apresentado um estudo prático que avalia a aplicabilidade de um Transformador como Discriminador de uma WGAN-GP, originando o modelo proposto, chamado de WGAN-GP-T. Três métodos XAI serão aplicados neste modelo com o objetivo de esclarecer seu processo de tomada de decisões. O modelo será utilizado como classificador de tráfego em um NIDS no ambiente SDN emulado. As próximas seções e subseções apresentarão um detalhamento do ambiente de execução, do sistema proposto e uma análise sobre os resultados apresentados.

7.1 Ambiente de Execução

O sistema foi pensado para ser utilizado no Plano de Aplicação do ambiente SDN. A Tabela 1 apresenta as configurações de *hardware* e *software* da máquina utilizada no desenvolvimento, treinamento e teste do sistema. Outo fator crucial para o seu desenvolvimento é o conjunto de dados, que possui as informações utilizadas na aprendizagem do modelo.

Tabela 1 – Configuração da máquina de desenvolvimento. Fonte: elaboração própria.

Sistema Operacional	Ubuntu 24.04.1 LTS
RAM	20 GB DDR4
Processador	AMD Ryzen™ 5 5500U @ 2,1 GHz × 12
Python	3.10.9
Tensorflow	2.11.0
Keras	2.11.0
Numpy	1.24.1
Pandas	1.5.3
Scikit-learn	1.2.1
Matplotlib	3.6.3
Bayesian-optimization	2.0.3
SHAP	0.46.0
Alibi (ALE)	0.9.6
LIME	0.2.0.1

7.1.1 Conjunto de Dados

Para o desenvolvimento de modelos de DL, é necessário um conjunto de dados representativo em relação ao problema abordado [3]. Para o ambiente SDN, existe uma escassez de conjuntos de dados representativos [2], possivelmente relacionado com o alto

custo de implantação da tecnologia. Uma solução adotada é o uso de emuladores de rede, que facilitam a criação de ambientes SDN e buscam manter a fidelidade em relação a ambientes reais. Além de representativos, é interessante que os dados sejam públicos, possibilitando a comparação de diferentes modelos para o mesmo ambiente.

O grupo de pesquisa Orion da Universidade Estadual de Londrina desenvolveu um conjunto de dados baseado em fluxos IP [42]. A ferramenta *Mininet* foi utilizada para emular a topologia de 6 *switches* e 140 *hosts*, *Floodlight* como controlador e *Scapy* como gerador de tráfego. Os ataques abordados foram *DDoS* e *Port Scan*, e as ferramentas utilizadas para suas reproduções foram *hping3* e *Scapy*, respectivamente. A Figura 16 apresenta em maior detalhe a disposição dos componentes utilizados no ambiente em questão.

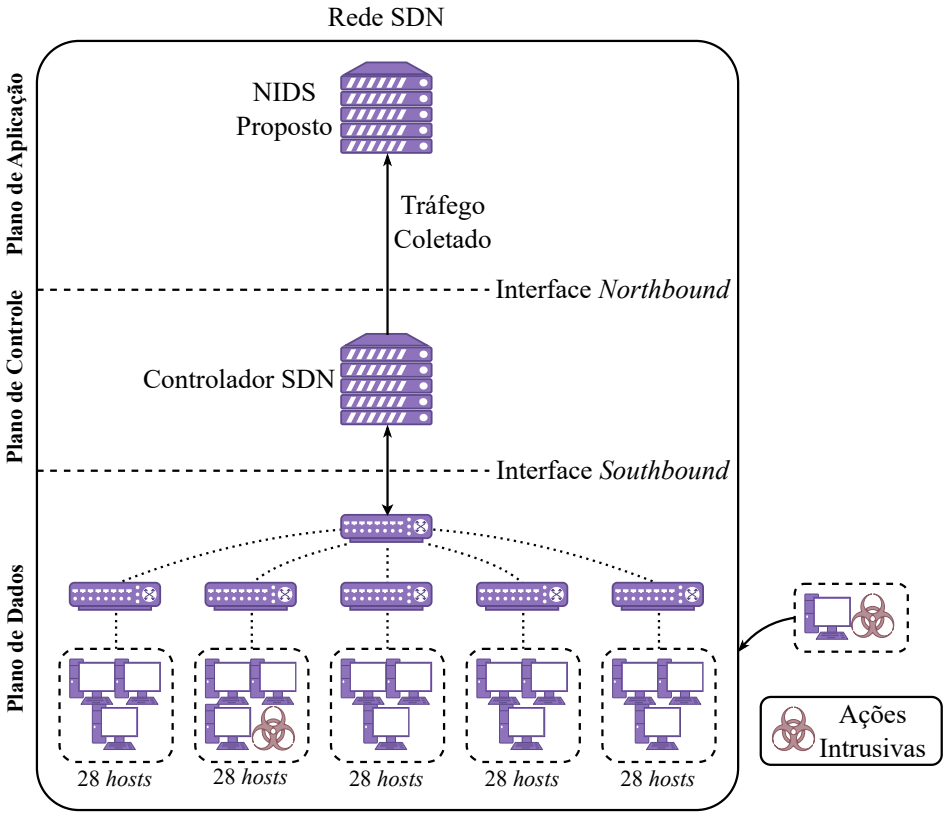

Figura 16 – Ambiente utilizado. Fonte: elaboração própria.

O conjunto de dados Orion possui três dias de tráfego rotulados em duas classes: normal e anômalo. O primeiro dia possui apenas o tráfego normal da rede, sem a presença de nenhum dos ataques. O segundo dia apresenta um ataque de *DDoS* das 9:15:00 às 10:30:00 e um ataque de *Port Scan* das 13:45:00 às 14:50:00. O terceiro dia contém um ataque de *Port Scan* das 11:20:00 às 12:40:00 e um ataque de *DDoS* das 16:35:00 às 17:50:00.

A base de dados apresenta seis características calculadas a cada segundo, totali-

zando 86.400 registros por dia. Estas características são entropia de IP de destino, entropia de porta de destino, entropia de IP de origem, entropia de porta de origem, total de bits e total de pacotes. A entropia utilizada é a de Shannon [96], sendo implementada com o objetivo de converter um conjunto de valores qualitativos em um único valor quantitativo, indicando a medida de dispersão ou concentração desses dados. A entropia aumentará com o acréscimo de novos valores qualitativos que seguem uma distribuição relativamente uniforme entre eles (dados igualmente dispersos), e diminuirá quando certos valores possuírem mais ocorrências que os demais (concentração dos dados).

No caso deste conjunto de dados, durante um ataque *DDoS* um *host* específico é sobrecarregado com múltiplas requisições, alterando a aparição de seu IP e porta nos dados de destino, consequentemente diminuindo a entropia dessas características. A entropia porta de origem aumentará, pois os agentes maliciosos utilizam portas aleatórias para o envio dos pacotes. No caso de um ataque *Port Scan*, o atacante envia requisições para as portas de um *host* específico para examiná-las, resultando no aumento da entropia de porta de destino.

A Entropia de Shannon é calculada conforme a Equação 7.1, onde x_t^A denota um histograma que contabiliza a frequência de ocorrência para cada valor possível do atributo qualitativo A durante o período t de análise. O conjunto x_t^A é composto por $\{x_1, x_2, ..., x_n\}$, onde cada x_i representa a contagem de ocorrências para o valor i específico. A variável S representa a soma de todas as ocorrências observadas para o atributo A no período analisado, sendo matematicamente expresso como $\sum_{i=1}^{n} x_i$.

$$H(x_t^A) = -\sum_{i=1}^{n} (\frac{x_i}{S}) \log_2 (\frac{x_i}{S}) \tag{7.1}$$

As Figuras 18, 19 e 20 contêm os valores das 6 características para os 3 dias nos 86.400 segundos coletados, respectivamente. A legenda desses três gráficos está na Figura 17.

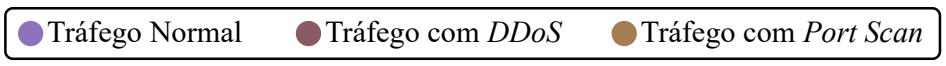

Figura 17 – Legenda dos gráficos das características do conjunto de dados. Fonte: elaboração própria.

Ruffo *et al.* (2025) [7] utilizam o mesmo conjunto de dados para o desenvolvimento de um NIDS no ambiente SDN emulado. Após a etapa de engenharia de características, os autores concluem que as características de total de bits e total de pacotes não contribuem para a aprendizagem do modelo. Isto é perceptível ao analisar as Figuras 19 e 20, reparando que as duas características em questão não apresentam variações de magnitude,

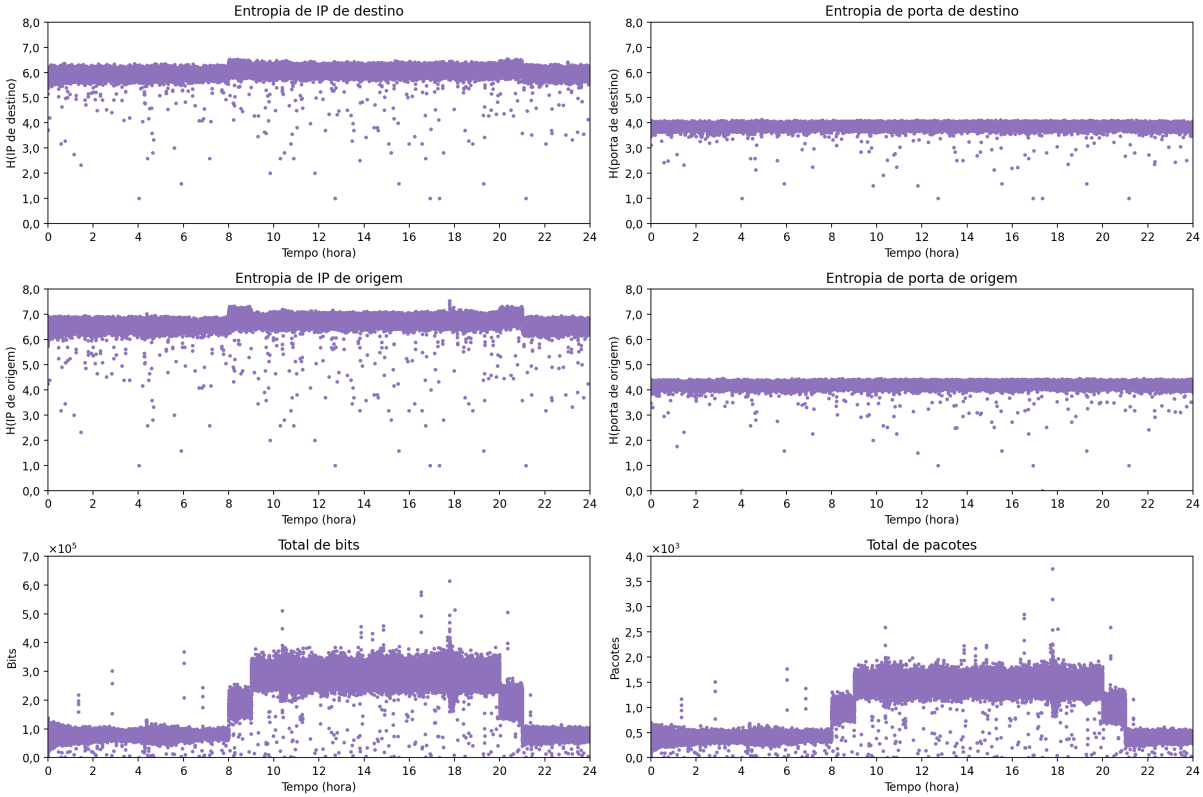

Figura 18 – Características do primeiro dia. Fonte: elaboração própria.

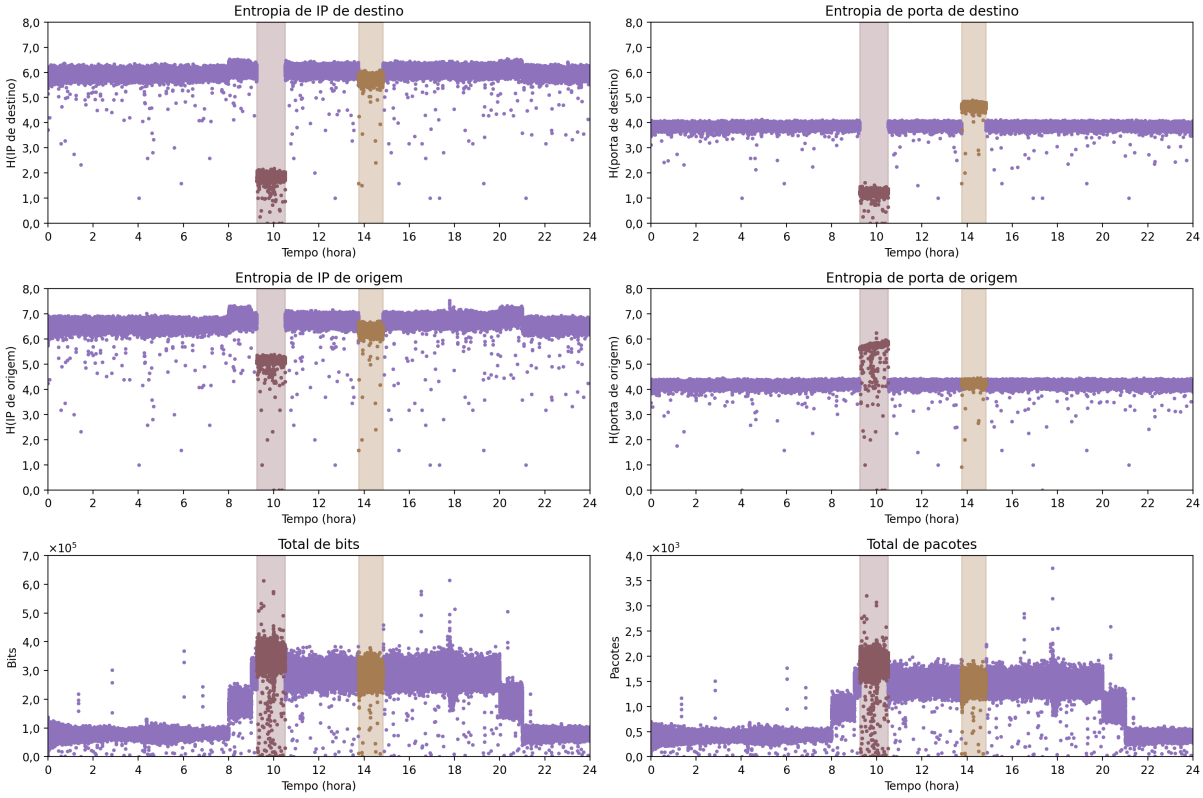

Figura 19 – Características do segundo dia. Fonte: elaboração própria.

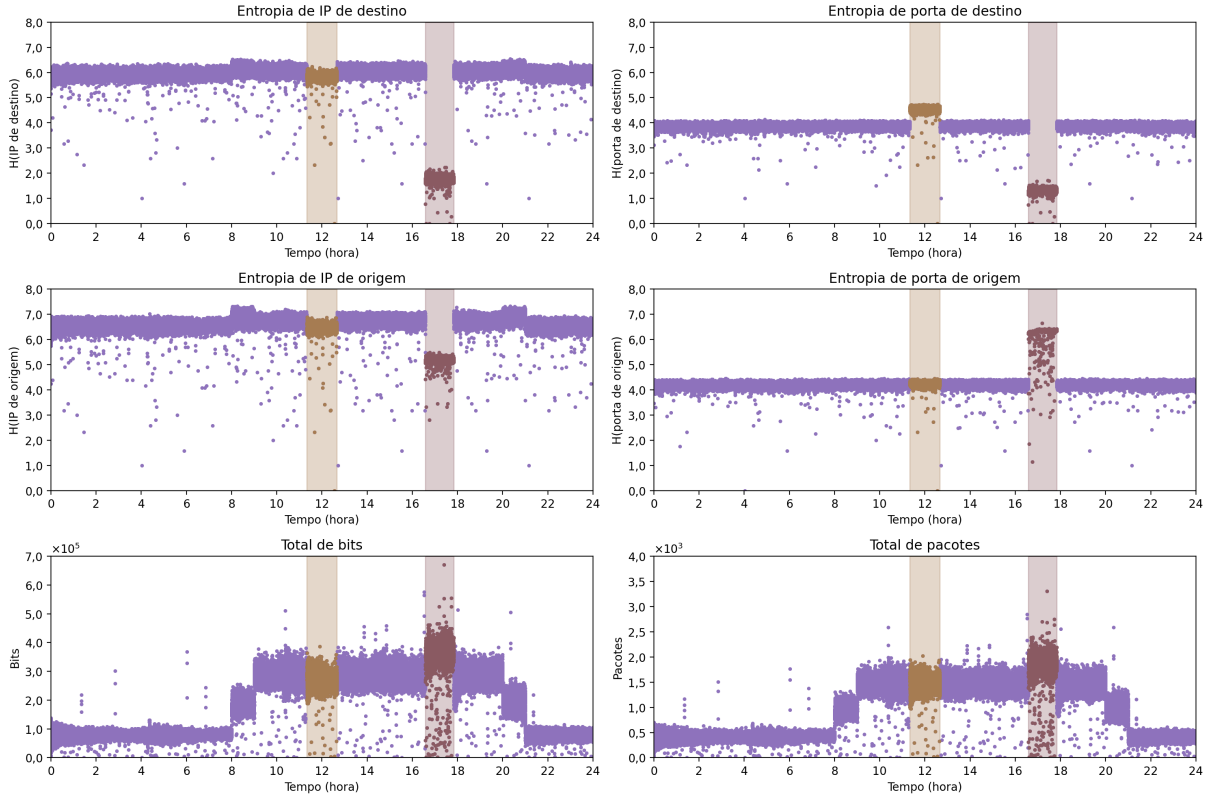

Figura 20 – Características do terceiro dia. Fonte: elaboração própria.

na presença de ataques, como as demais características. Seguindo a análise feita pelos autores, este trabalho também utilizará apenas as características de entropia.

7.2 Sistema de Detecção de Intrusão de Redes

A Figura 21 ilustra o funcionamento interno do NIDS proposto, exibindo a divisão de funções, pontuação produzida e possível alerta gerado. O processo se inicia com a coleta e pré-processamento do tráfego que será avaliado pelo modelo proposto e classificado em relação aos limiares estabelecidos. Caso o tráfego analisado seja classificado como normal, nada é feito, mas caso seja como anômalo, um alerta é enviado para o gerente de rede.

7.2.1 Coleta e Pré-processamento de Tráfego

A primeira função realizada pelo NIDS é a coleta de tráfego. Ela se inicia com uma requisição enviada, por meio da interface *northbound*, para o controlador. Este, por sua vez, retorna para o NIDS as informações selecionadas. Não há uma regra que defina o intervalo correto para estas coletas, ele apenas deve ser rápido o suficiente para evitar danos à rede. O sistema proposto em [5] coleta tráfego a cada segundo, obtendo sucesso na detecção de intrusões. O intervalo de coleta e pré-processamento escolhido para este trabalho será de um segundo.

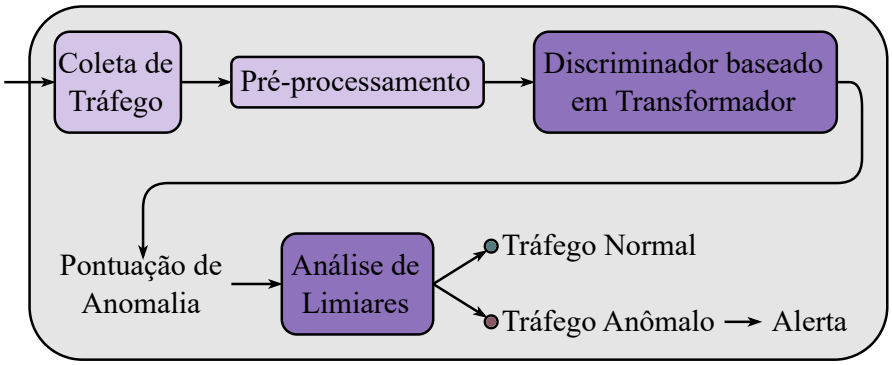

Figura 21 – NIDS proposto. Fonte: elaboração própria.

Para auxiliar na aprendizagem do modelo proposto, os dados coletados devem ser pré-processados. O primeiro passo é garantir que os valores coletados sejam apenas numéricos, removendo valores indesejados que possivelmente surgiram de falhas na coleta do controlador. Com os valores corretos separados, será feito o cálculo das entropias, como explicado na Subseção 7.1.1. As entropias de cada característica resultam em diferentes intervalos numéricos, como pode ser observado nos eixos y da Figura 18. Essa diferença de intervalos dificulta a aprendizagem do modelo, pois os dados inseridos nele não seguem a mesma escala. Este problema é resolvido por meio da normalização, a próxima etapa do pré-processamento.

O objetivo da normalização é manter os valores desejados dentro de um intervalo específico. O método de normalização utilizado neste trabalho é o *MinMax*, que normaliza os valores entre 0 e 1 a partir de seus mínimos e máximos. Espera-se que a amostra sintética de G esteja neste mesmo intervalo, isto implica no uso da função de ativação *sigmoid*, que está relacionada com propriedades do seno e garante que a saída do neurônio estará entre 0 e 1. A função *MinMaxScaler* fornecida pela biblioteca *scikit-learn* foi utilizada neste trabalho. Segundo a definição em [97], esse escalador pode ser definido pela seguinte equação:

$$X_{escalado} = \frac{X_i - \min(X)}{\max(X) - \min(X)}, \tag{7.2}$$

onde X_i é o i-ésimo elemento do conjunto de dados X, e as funções *min()* e *max()* retornam o maior e menor valor, respectivamente, presentes no conjunto em questão. Após esses passos, os dados estão prontos para serem inseridos no modelo de DL.

7.2.2 Modelo WGAN-GP-T

O modelo proposto utiliza o Transformador como Discriminador da WGAN-GP. Para realizar esta combinação, a saída linear da última camada densa do Transformador é utilizada como a pontuação de D. As funções de perda utilizadas continuam sendo as da

WGAN-GP, para manter o uso da distância de *Wasserstein*. O modelo é treinado apenas com os dados normais (primeiro dia) do conjunto de dados. O objetivo, neste caso, é que D aprenda a distribuição dos dados normais, produzindo pontuações semelhantes para eles. Espera-se que a análise de amostras anômalas produza pontuações que destoem das normais. É necessário definir limites que separem pontuações normais das anômalas.

Algumas partes da arquitetura original do Transformador foram removidas por não serem úteis para o caso de uso deste trabalho. A última camada *softmax* foi removida para possibilitar o uso dos valores numéricos produzidos pelo Transformador. As probabilidades produzidas pela *softmax* não seriam usadas, pois as funções de distância da WGAN-GP exigem as pontuações produzidas pelo modelo. A etapa de Incorporação de Entrada também não foi utilizada, porque ela está relacionada com a transformação de segmentos de palavras em valores numéricos. Os dados utilizados neste trabalho já são numéricos e podem, após normalização, ser inseridos diretamente no modelo. Outra etapa removida foi a de Codificação Posicional. As características analisadas serão inseridas sempre na mesma ordem, portanto não é necessária uma codificação que atribua diferentes pesos para cada valor de entrada com base em qual posição ele se encontra.

7.2.3 Limiares de Detecção

Para estabelecer os limiares de detecção, foram utilizados conceitos da distribuição normal, também conhecido como distribuição Gaussiana. Esta escolha se justifica pelo Teorema do Limite Central [98], que estabelece que a distribuição de médias amostrais tende a uma distribuição normal conforme o tamanho da amostra aumenta. Considerando que a pontuação de anomalia produzida por D para dados normais deve seguir um padrão consistente, foram definidos dois limiares baseados na média (μ) e no desvio-padrão (σ) dessas pontuações: um limite superior estabelecido como $\mu + 2\sigma$ e um limite inferior como $\mu - 2\sigma$. Foram utilizados dois limiares, pois não é garantido que o modelo produzirá pontuações maiores para amostras anômalas, somente espera-se que elas difiram das normais.

Esta abordagem é fundamentada no fato de que, em uma distribuição normal, aproximadamente 95,4% dos dados se encontram dentro do intervalo de dois desvios-padrão da média, como pode ser observado na Figura 22. Também foram utilizados como base os experimentos e descrições realizados por Boppana *et al.* [99], que analisaram diferentes coeficientes (1, 2 e 3) na multiplicação do desvio-padrão para obtenção dos limiares. As pontuações que ultrapassem estes limiares são classificadas como anomalias, pois representam comportamentos que desviam, além da tolerância pré-definida, do padrão normal aprendido pelo modelo.

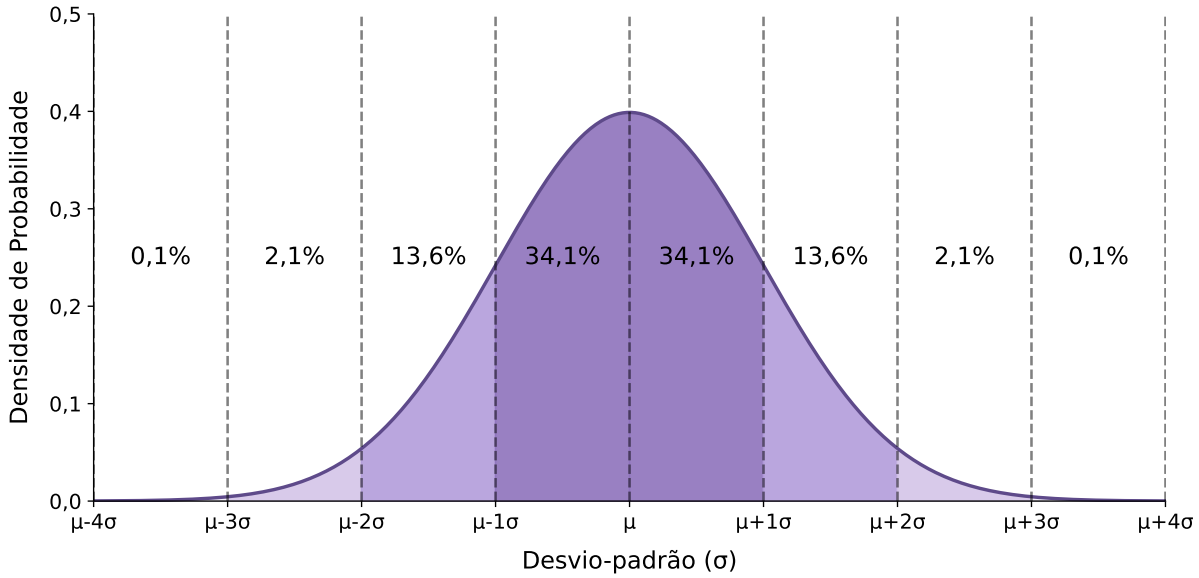

Figura 22 – Distribuição Normal. Fonte: elaboração própria.

7.2.4 Resultados da Otimização de Hiperparâmetros

Para otimizar os hiperparâmetros do modelo proposto, foi utilizada a Otimização Bayesiana, como especificado na Subseção 2.4.4. A biblioteca *bayesian-optimization* [100] foi utilizada nesta etapa. Este método é eficiente em encontrar configurações próximas do ótimo global com um número reduzido de iterações quando comparada a outras técnicas, como busca em grade ou busca aleatória. O processo de otimização foi executado por 20 iterações, utilizando 10 divisões para avaliar o desempenho de cada configuração testada. A métrica utilizada para guiar a otimização foi o MCC, por ser uma métrica robusta que considera todos os valores da matriz de confusão. A Tabela 2 apresenta os hiperparâmetros otimizados e seus respectivos valores encontrados pela otimização, que foram utilizados no treinamento final do modelo. Alguns hiperparâmetros da WGAN-GP e do Transformador tiveram seus valores definidos com base nos respectivos artigos que propuseram essas arquiteturas. Estes hiperparâmetros foram algoritmo de otimização (Adam), função de ativação de G (ReLU), taxa de aprendizagem (0,0001 para G e a própria do Transformador para D), função de perda (próprias da WGAN-GP), coeficiente da penalidade de gradiente (10) e passos extras de D (5).

7.3 Resultados

O modelo final foi testado com o segundo e terceiro dias do conjunto de dados, totalizando 172.800 amostras analisadas, sendo 155.100 normais e 17.700 anômalas. Das anômalas, 9.000 são de ataques *DDoS* e 8.700 de *Port Scan*.

Tabela 2 – Hiperparâmetros após otimização. Fonte: elaboração própria.

Hiperparâmetros	**Valor**
Número de épocas	26
Tamanho de lote	125
Tamanho do espaço latente	111
Camadas ocultas G	3
Neurônios por camada G	121
Taxa de abandono D	0,2929
Número de camadas de Transformador	2
Número de cabeças de atenção	2
Neurônios da camada de Alimentação Direta	128

7.3.1 Matriz de Confusão e suas Métricas

A Figura 23 apresenta a matriz de confusão para os dias de teste. Foram 17.628 verdadeiros positivos, 154.480 verdadeiros negativos, 620 falsos positivos e 72 falsos negativos, totalizando 172.108 classificações corretas e 692 classificações erradas. A partir desses valores foram calculadas as métricas apresentadas na Tabela 3.

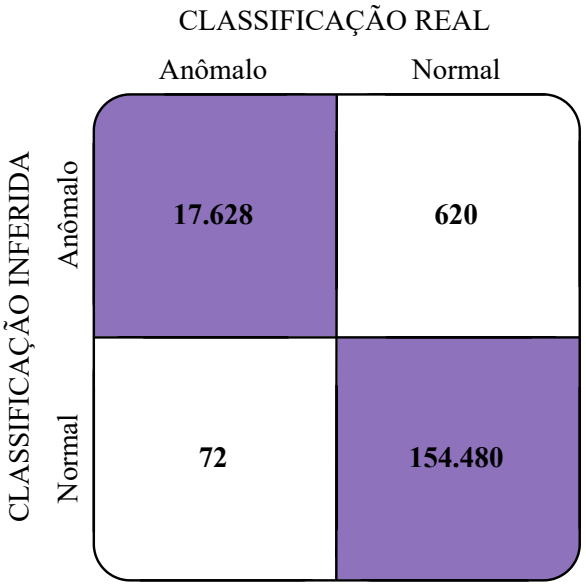

Figura 23 – Resultado da matriz de confusão. Fonte: elaboração própria.

A Figura 24 exibe a curva ROC e o valor AUROC (0,9960) obtido. A escala dos eixos da Figura foi alterada para facilitar na visualização da curvatura, indo de [0; 1] nos dois eixos para [0; 0,2] no eixo x e [0,8; 1] no eixo y. É possível perceber que a curva obtida se aproximou das bordas esquerda e superior, resultando no valor AUROC próximo à 1. Esses resultados levam a concluir que o modelo conseguiu diferenciar classes benignas das malignas.

Tabela 3 – Métricas de desempenho. Fonte: elaboração própria.

Métricas	**Valor**
Acurácia	0,9960
Precisão	0,9660
Revocação	0,9959
F1-*score*	0,9807
MCC	0,9787

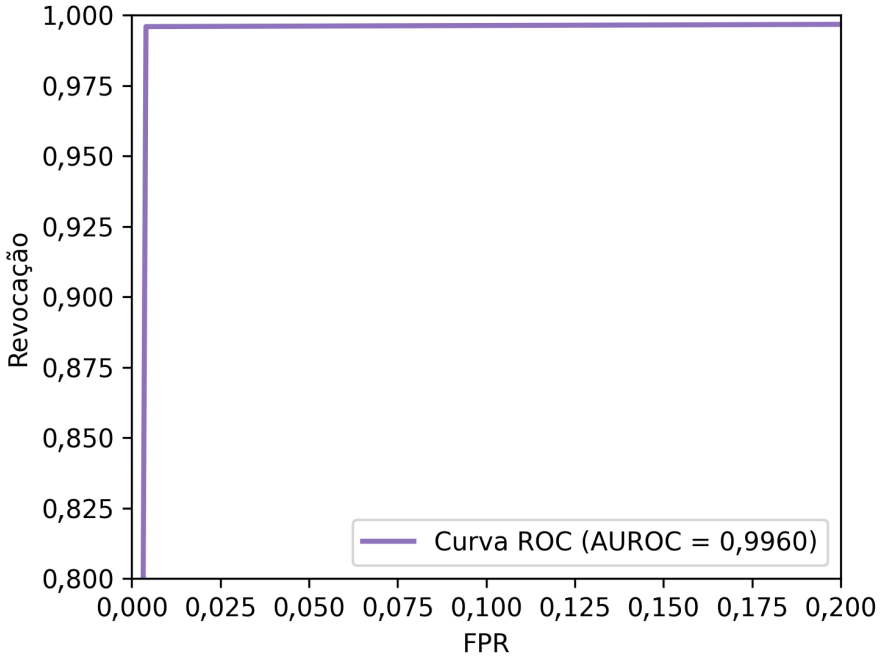

Figura 24 – Curva ROC e valor AUROC. Fonte: elaboração própria.

7.3.2 Métodos XAI

O modelo proposto foi submetido à análise de métodos XAI para elucidar seu processo de tomada de decisão. O foco dos métodos utilizados está na explicabilidade, mostrando como cada característica de entrada, para cada tipo de tráfego, influencia na classificação do modelo.

7.3.2.1 SHAP

A Figura 25 apresenta os resultados da saída SHAP para a análise de 900 amostras aleatórias dos dias de teste, sendo 300 de dados normais, 300 de *DDoS* e 300 de *Port Scan*. Os resultados foram obtidos utilizando a função *KernelExplainer* da biblioteca *shap* [85]. O método *Kernel* SHAP combina noções do LIME com SHAP, utilizando aproximações lineares locais e conceitos da teoria dos jogos para estimar os valores de Shapley, já que a computação exata desses valores é desafiadora devido ao alto custo computacional [85]. A

Figura 25 (a) demonstra que valores maiores da entropia de porta de destino influenciaram negativamente o modelo, enquanto valores menores desta mesma entropia influenciaram positivamente. Valores menores da entropia de IP de destino e valores maiores da entropia de porta de origem influenciaram negativamente nas decisões do modelo. A Figura 25 (b) mostra que as características que mais influenciaram o modelo, independente de ser positiva ou negativamente, foram entropia de porta de destino e entropia de IP de destino.

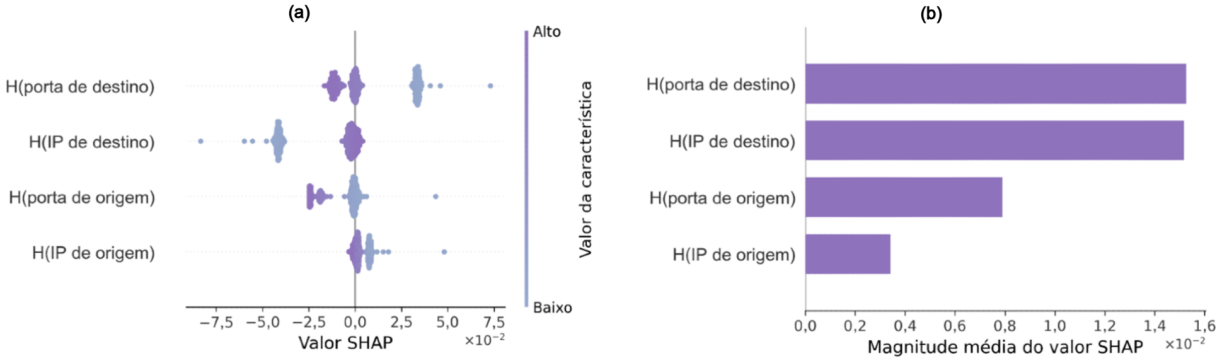

Figura 25 – Saída SHAP para a análise de 900 amostras. Fonte: elaboração própria.

A Figura 26 traz uma análise mais detalhada acerca das saídas SHAP para cada tipo de tráfego. No tráfego normal (a), não há nenhuma característica que se destaca no quesito de influência, mas os valores maiores da entropia de IP de destino e os menores das demais entropias estão mais a direita. No tráfego *DDoS* (b), todas as amostras analisadas da entropia de IP de destino influenciaram negativamente na predição do modelo. Pelo contrário, todas as amostras da entropia de porta de destino influenciaram positivamente. Já no tráfego *Port Scan* (c), não houve nenhum destaque em relação à influência positiva, sendo que a característica mais a direita foi a entropia de IP de origem, especificamente os valores menores. Analisando as influências negativas para este tipo de tráfego, destacam-se as entropias de destino, que estão, totalmente à esquerda.

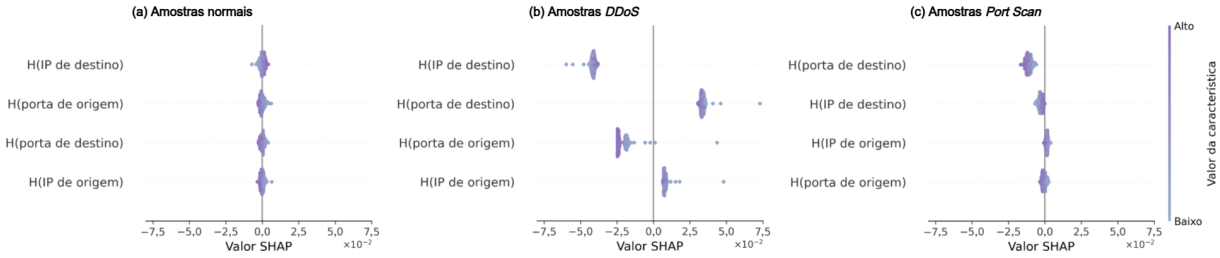

Figura 26 – Saída SHAP separada por amostras normais (a), *DDoS* (b) e *Port Scan* (c). Fonte: elaboração própria.

7.3.2.2 ALE

A Figura 27 exibe os resultados da saída ALE para a análise dos dados do segundo e terceiro dias. Os resultados foram obtidos utilizando a função *ALE* da biblioteca *alibi*

[101]. Os resultados exibem as diferentes distribuições dos valores de cada característica. As que mais influenciaram positivamente o modelo foram os menores valores da entropia de porta de destino e da entropia de porta de origem. Os maiores valores dessas entropias foram os que mais influenciaram negativamente as predições, juntamente dos menores valores da entropia de IP de destino. Analisando as saídas ALE, também é possível observar as diferentes concentrações dos valores dos dados, onde a entropia de IP de origem apresenta intervalos mais concentrados em relação à entropia de porta de destino, por exemplo.

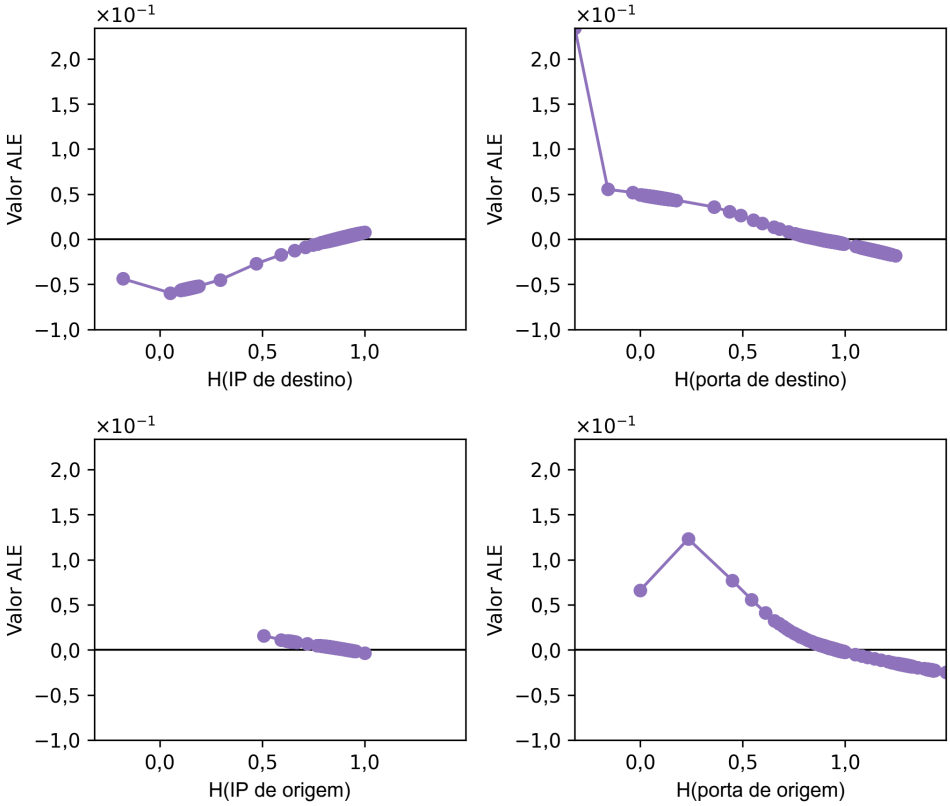

Figura 27 – Saída ALE para as quatro características. Fonte: elaboração própria.

7.3.2.3 LIME

Para as próximas análises, foi utilizada a função *LimeTabularExplainer* da biblioteca Lime [88]. As três amostras utilizadas foram selecionadas aleatoriamente dentro de cada tipo de tráfego. A Figura 28 mostra a análise feita sobre uma amostra de tráfego normal. Para esta amostra, o modelo acertou a classificação, indicando-a como benigna. Também é indicado como cada característica influenciou para decisão, sendo a entropia de porta de destino e a de porta de origem as que mais influenciaram. Neste caso, todas as entropias influenciaram positivamente, mas o destaque foi para as relacionadas as portas.

A Figura 29 exibe a análise feita sobre uma amostra de tráfego contendo *DDoS*. O modelo acertou na predição, indicando que a característica que mais auxiliou foi a

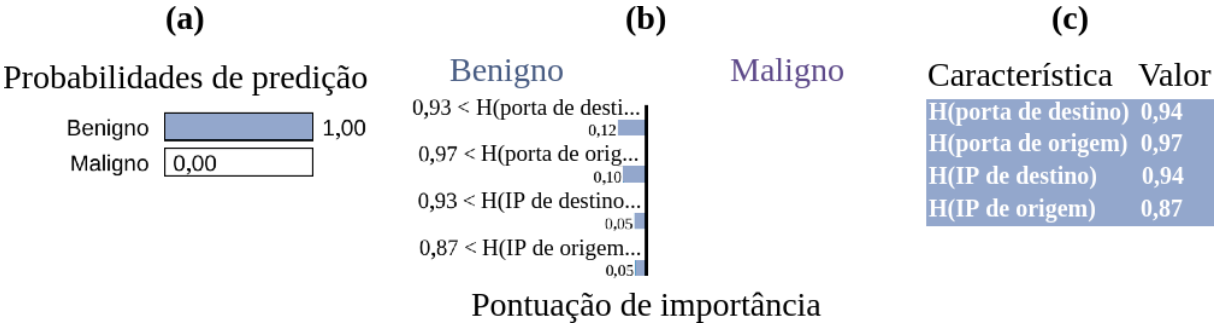

Figura 28 – Saída LIME para uma amostra normal. Fonte: elaboração própria.

entropia de IP de destino, assim como as de porta de destino e a de origem. Nenhuma das características influenciou negativamente na predição, e seus respectivos valores podem ser observados na parte direita da figura.

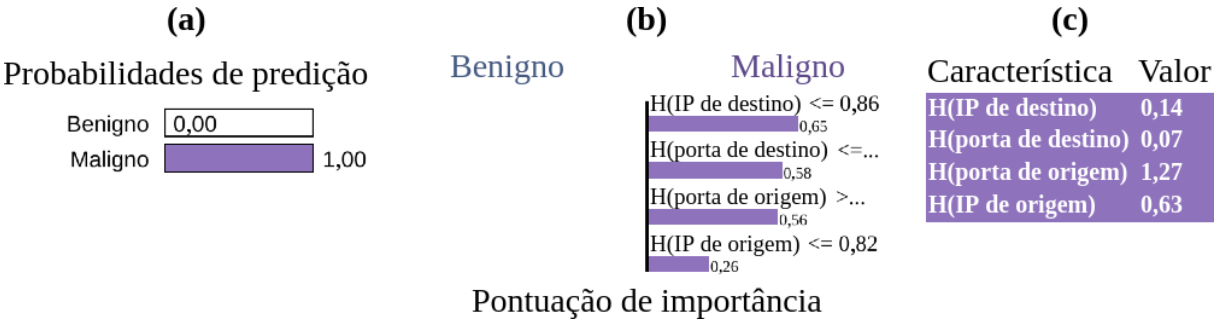

Figura 29 – Saída LIME para uma amostra *DDoS*. Fonte: elaboração própria.

A Figura 30 contém os detalhes de uma análise feita sobre uma amostra de tráfego contendo *Port Scan*. O modelo acertou na predição, tendo como maior influência a entropia de porta de destino e a de IP de origem. Houve duas características que influenciaram negativamente, sendo as entropias de IP de destino e porta de origem.

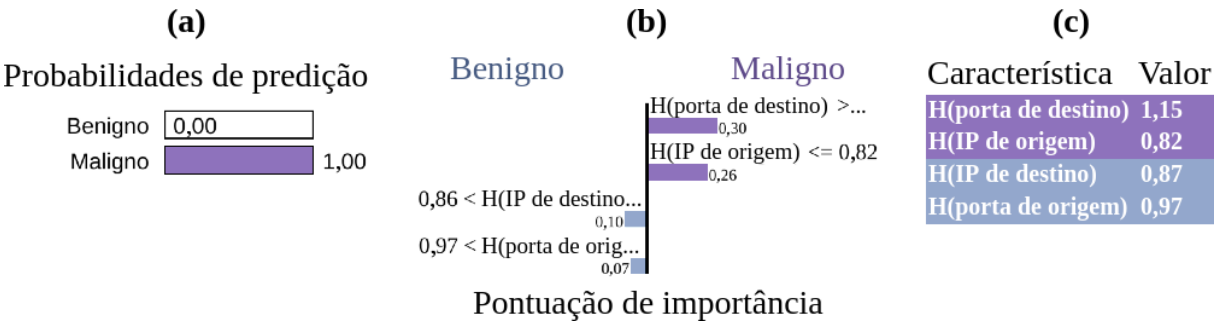

Figura 30 – Saída LIME para uma amostra *Port Scan*. Fonte: elaboração própria.

Nas três análises utilizando LIME, as duas únicas características que apareceram como mais influentes em cada caso foram as entropias de porta de destino e IP de destino. Esta observação condiz com o que foi apresentado nos resultados dos métodos SHAP e ALE.

7.3.3 Discussão

O modelo WGAN-GP-T foi capaz de detectar os ataques, apresentando todas as métricas de avaliação acima de 96%. No entanto, o modelo errou a classificação de 692 amostras, sendo a maioria relacionada aos falsos positivos (620). Isto é um indicativo de que o modelo não conseguiu aprender totalmente a distribuição dos dados normais. Esta falha na aprendizagem está relacionada com a dificuldade de equilibrar o treinamento não supervisionado das duas arquiteturas do estado da arte. Essa dificuldade de equilíbrio pode ser visualizada na Figura 31, que mostra a instabilidade dos valores de perda de cada rede em relação às épocas de treinamento. Além disso, o propósito original de cada arquitetura difere do uso empregado neste trabalho, o que também interfere na aprendizagem do modelo que combina ambas.

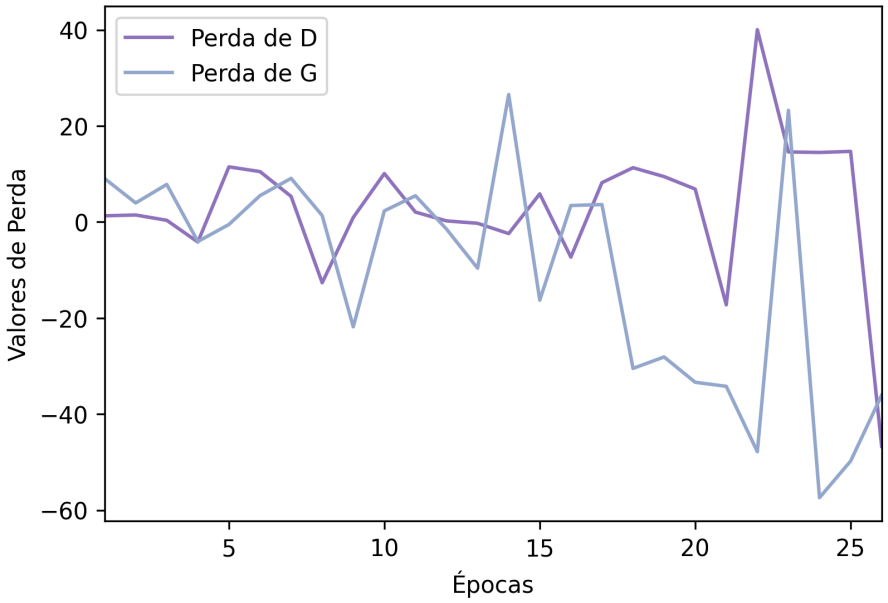

Figura 31 – Valores de perda por época. Fonte: elaboração própria.

Outra análise em relação aos resultados exibidos envolve os métodos XAI utilizados. Os três métodos indicam, de modo geral, que as duas características mais influentes são a entropia de porta e IP de destino, respectivamente. Elas influenciam de diferentes maneiras para cada tipo de tráfego. Este resultado está coerente com o que poder ser visualizado nas Figuras 19 e 20, onde as entropias que mais destacam os ataques *DDoS* são as de destino, e a que mais destaca os ataques *Port Scan* é a de porta de destino. Estas informações também estão de acordo com o funcionamento desses ataques, pois o *DDoS* focará em um conjunto de IPs e portas de destino específicos, reduzindo a entropia deles; já o *Port Scan* analisará um conjunto de diversas portas de destino, aumentando esta entropia.

Após sintetizar as análises realizadas com o auxílio das figuras SHAP, ALE e LIME, é possível dizer que a característica mais influente na detecção das anomalias é

a entropia da porta de destino. Este resultado indica que seria possível desenvolver um modelo utilizando somente esta característica. A redução do número de características processadas pelo modelo auxilia na redução do seu custo de processamento, reduzindo o tempo de inferência e, consequentemente, da detecção das anomalias. A desvantagem dessa redução é que menos informações estarão disponíveis na aprendizagem do modelo, reduzindo seu escopo de funcionamento.

8 CONCLUSÃO

Neste trabalho foi apresentado um estudo sobre a aplicabilidade da combinação da arquitetura do Transformador com a arquitetura WGAN-GP no desenvolvimento de um NIDS para o ambiente SDN emulado. Além disso, também foram utilizados métodos XAI no modelo final, buscando compreender como ele relacionava as características de entrada. O processamento do NIDS proposto começa com a coleta e pré-processamento do tráfego, os dados resultantes dessa primeira etapa são passados para um classificador de tráfego. O Discriminador da WGAN-GP, implementado com um Transformador, foi desenvolvido para classificar as amostras, utilizando sua saída como pontuação de anomalia e contando com limiares definidos a partir de conceitos da distribuição normal, configurando um modelo não supervisionado.

O sistema proposto foi treinado e testado utilizando uma base de dados sintética, coletada por meio de emulações SDN. Este conjunto de dados conta com um dia sem ataque, usado para o treino, e dois dias com ataques de *DDoS* e *Port Scan*, usados para o teste. Esta base de dados também apresenta o conceito das entropias, que auxiliam na detecção destes ataques. O desempenho do classificador binário foi medido utilizando-se a matriz de confusão e as métricas que derivam dela, sendo: Acurácia, Precisão, Revocação, F1-*score*, MCC, ROC e AUROC.

O modelo WGAN-GP-T foi capaz de detectar os ataques, apresentando resultados como 99,60% de Acurácia, 96,60% de Precisão, 99,59% de Revocação, 98,07% de F1-*score*, 97,87% de MCC e 99,60% de AUROC. Entretanto, o modelo errou na classificação de 692 amostras, tendo um número notável de falsos positivos (620). Estes valores elevados de falso positivo indicam que D não foi capaz de aprender a distribuição dos dados normais, então classifica diversas instâncias normais como anômalas. Esta falha na aprendizagem está, possivelmente, relacionada com a instabilidade do modelo. Por se tratar da combinação de duas arquiteturas complexas, não é uma tarefa trivial balancear a aprendizagem de ambas, proporcionando um treinamento estável. Outro ponto a ser considerado é que ambas as arquiteturas não foram desenvolvidas especificamente para a detecção de anomalias, então elas funcionam para este tipo de problema através de adaptações no seu funcionamento.

Embora o modelo tenha demonstrado capacidade de identificar anomalias no tráfego SDN, ele ainda apresenta possíveis pontos de melhoria, principalmente relacionado à taxa de falsos positivos. Esse resultado decorre, principalmente, da dificuldade de estabilizar o treinamento do sistema híbrido WGAN-GP-T, evidenciando os desafios em combinar arquiteturas complexas de forma estável. Embora as arquiteturas WGAN-GP e Transformador sejam reconhecidamente eficazes em diversas tarefas, como geração de

imagens e tradução de textos, que envolvem aprendizado profundo, elas não foram projetadas especificamente para detecção de anomalias, exigindo adaptações para seu uso nessa aplicação.

As vantagens individuais das arquiteturas WGAN-GP e Transformador também foram evidenciadas neste trabalho. A WGAN-GP demonstra seu potencial ao estabilizar o treinamento em ambientes adversariais, sendo capaz de modelar distribuições complexas de dados, mesmo em cenários com variabilidade. A arquitetura do Transformador se destaca na identificação de padrões ao longo das sequências dos dados, graças à sua capacidade de capturar dependências de longo alcance e processar os dados de forma paralela. Apesar dos desafios de estabilização e adaptação da combinação dessas arquiteturas ao contexto de detecção de anomalias, suas propriedades intrínsecas mostram potencial para aplicações futuras, especialmente se exploradas com ajustes específicos para o problema em questão.

Como trabalhos futuros destaca-se a busca de novas formas para estabilizar o treinamento do modelo, por meio de alterações em funções de custo ou componentes da arquitetura, por exemplo. Estas mudanças visam sincronizar de forma mais eficiente as duas arquiteturas do estado da arte. Outro objetivo futuro é treinar e testar o modelo com diferentes bases de dados, que contenham um padrão de tráfego diferente e tipos de ataques mais variados. Outra possível alteração envolvendo os dados seria a utilização de uma única característica, sendo ela a entropia de porta de destino, a mais influente segundo as análises XAI. Isto reduziria o custo computacional do modelo, possivelmente reduzindo seu tempo de inferência. Outra melhoria envolvendo os métodos XAI, seria a utilização de algum método de subamostragem para selecionar as instâncias a serem analisadas pelo SHAP e LIME. Isto garantiria a representatividade dos dados selecionados, evitando instâncias discrepantes. A adição de outras métricas de avaliação é outra possível melhoria, como tempo de treinamento, tempo de inferência e complexidade do modelo, sendo estas informações que alteram o desempenho de um NIDS real. O modelo também poderia utilizar métricas que avaliam o quão bem ele consegue distanciar amostras anômalas das normais. Além disso, o desempenho do modelo proposto poderia ser comparado com outros trabalhos semelhantes presentes em artigos científicos.

REFERÊNCIAS

[1] VASWANI, A. et al. *Attention Is All You Need*. 2023. Disponível em: <https://doi.org/10.48550/arXiv.1706.03762>.

[2] da Silva Ruffo, V. G. et al. Anomaly and intrusion detection using deep learning for software-defined networks: A survey. *Expert Systems with Applications*, v. 256, p. 124982, 2024. ISSN 0957-4174. Disponível em: <https://doi.org/10.1016/j.eswa.2024.124982>.

[3] YANG, Z. et al. A systematic literature review of methods and datasets for anomaly-based network intrusion detection. *Computers Security*, v. 116, p. 102675, 2022. ISSN 0167-4048. Disponível em: <https://doi.org/10.1016/j.cose.2022.102675>.

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

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

[6] SHARAFALDIN, I.; Habibi Lashkari, A.; GHORBANI, A. A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: INSTICC. *Proceedings of the 4th International Conference on Information Systems Security and Privacy - ICISSP*. SciTePress, 2018. p. 108–116. ISBN 978-989-758-282-0. ISSN 2184-4356. Disponível em: <https://doi.org/10.5220/0006639801080116>.

[7] da Silva Ruffo, V. G. et al. Generative adversarial networks to detect intrusion and anomaly in ip flow-based networks. *Future Generation Computer Systems*, v. 163, p. 107531, 2025. ISSN 0167-739X. Disponível em: <https://doi.org/10.1016/j.future.2024.107531>.

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

[9] LENT, D. M. B. et al. An unsupervised generative adversarial network system to detect ddos attacks in sdn. *IEEE Access*, v. 12, p. 70690–70706, 2024. Disponível em: <https://doi.org/10.1109/ACCESS.2024.3402069>.

[10] HAN, Z. et al. A review of deep learning models for time series prediction. *IEEE Sensors Journal*, v. 21, n. 6, p. 7833–7848, 2021. Disponível em: <https://doi.org/10.1109/JSEN.2019.2923982>.

[11] GOODFELLOW, I. J. et al. *Generative Adversarial Networks*. 2014. Disponível em: <https://doi.org/10.48550/arXiv.1406.2661>.

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

[13] GUI, J. et al. A review on generative adversarial networks: Algorithms, theory, and applications. *IEEE Transactions on Knowledge and Data Engineering*, v. 35, n. 4, p. 3313–3332, 2023. Disponível em: <https://doi.org/10.1109/TKDE.2021.3130191>.

[14] SHIN, A.; ISHII, M.; NARIHIRA, T. *Perspectives and Prospects on Transformer Architecture for Cross-Modal Tasks with Language and Vision*. 2021. Disponível em: <https://doi.org/10.48550/arXiv.2103.04037>.

[15] ISLAM, S. et al. A comprehensive survey on applications of transformers for deep learning tasks. *Expert Systems with Applications*, v. 241, p. 122666, 2024. ISSN 0957-4174. Disponível em: <https://doi.org/10.1016/j.eswa.2023.122666>.

[16] LIM, W. et al. Future of generative adversarial networks (gan) for anomaly detection in network security: A review. *Computers Security*, v. 139, p. 103733, 2024. ISSN 0167-4048. Disponível em: <https://doi.org/10.1016/j.cose.2024.103733>.

[17] KANG, H.; KANG, P. Transformer-based multivariate time series anomaly detection using inter-variable attention mechanism. *Knowledge-Based Systems*, v. 290, p. 111507, 2024. ISSN 0950-7051. Disponível em: <https://doi.org/10.1016/j.knosys.2024.111507>.

[18] ALI, S. et al. Explainable artificial intelligence (xai): What we know and what is left to attain trustworthy artificial intelligence. *Information Fusion*, v. 99, p. 101805, 2023. ISSN 1566-2535. Disponível em: <https://doi.org/10.1016/j.inffus.2023.101805>.

[19] KOZIK, R. et al. When explainability turns into a threat - using xai to fool a fake news detection method. *Computers Security*, v. 137, p. 103599, 2024. ISSN 0167-4048. Disponível em: <https://doi.org/10.1016/j.cose.2023.103599>.

[20] TANENBAUM, A. S.; WETHERALL, D. J. *Computer Networks*. 5th. ed. USA: Prentice Hall Press, 2010. ISBN 0132126958.

[21] PHAN, T. V.; BAUSCHERT, T. Deepair: Deep reinforcement learning for adaptive intrusion response in software-defined networks. *IEEE Transactions on Network and Service Management*, v. 19, n. 3, p. 2207–2218, 2022. Disponível em: <https://doi.org/10.1109/TNSM.2022.3158468>.

[22] 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*, v. 10, p. 73229–73242, 2022. Disponível em: <https://doi.org/10.1109/ACCESS.2022.3190008>.

[23] BUZZIO-GARCíA, J. et al. Exploring traffic patterns through network programmability: Introducing sdnflow, a comprehensive openflow-based statistics dataset for attack detection. *IEEE Access*, v. 12, p. 42163–42180, 2024. Disponível em: <https://doi.org/10.1109/ACCESS.2024.3378271>.

[24] BHARDWAJ, A. et al. Network intrusion detection in software defined networking with self-organized constraint-based intelligent learning framework. *Measurement: Sensors*, v. 24, p. 100580, 2022. ISSN 2665-9174. Disponível em: <https://doi.org/10.1016/j.measen.2022.100580>.

[25] 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*, v. 214, p. 103622, 2023. ISSN 1084-8045. Disponível em: <https://doi.org/10.1016/j.jnca.2023.103622>.

[26] SCARANTI, G. F. et al. Unsupervised online anomaly detection in software defined network environments. *Expert Systems with Applications*, v. 191, p. 116225, 2022. ISSN 0957-4174. Disponível em: <https://doi.org/10.1016/j.eswa.2021.116225>.

[27] FOULADI, R. F.; ERMIş, O.; ANARIM, E. A ddos attack detection and countermeasure scheme based on dwt and auto-encoder neural network for sdn. *Computer Networks*, v. 214, p. 109140, 2022. ISSN 1389-1286. Disponível em: <https://doi.org/10.1016/j.comnet.2022.109140>.

[28] JANABI, A. H.; KANAKIS, T.; JOHNSON, M. Convolutional neural network based algorithm for early warning proactive system security in software defined networks. *IEEE Access*, v. 10, p. 14301–14310, 2022. Disponível em: <https://doi.org/10.1109/ACCESS.2022.3148134>.

[29] XU, L. et al. Tgan-ad: Transformer-based gan for anomaly detection of time series data. *Applied Sciences*, v. 12, n. 16, 2022. ISSN 2076-3417. Disponível em: <https://doi.org/10.3390/app12168085>.

[30] HUANG, S.; LEI, K. Igan-ids: An imbalanced generative adversarial network towards intrusion detection system in ad-hoc networks. *Ad Hoc Networks*, v. 105, p. 102177, 2020. ISSN 1570-8705. Disponível em: <https://doi.org/10.1016/j.adhoc.2020.102177>.

[31] HAMZA, A. et al. Detecting volumetric attacks on lot devices via sdn-based monitoring of mud activity. In: *Proceedings of the 2019 ACM Symposium on SDN Research*. New York, NY, USA: Association for Computing Machinery, 2019. (SOSR '19), p. 36–48. ISBN 9781450367103. Disponível em: <https://doi.org/10.1145/3314148.3314352>.

[32] ALLIANCE ethernet. *2024 Ethernet Roadmap - Ethernet Alliance — ethernetalliance.org*. 2024. [Acessado em 07-10-2024]. Disponível em: <https://ethernetalliance.org/technology/ethernet-roadmap/>.

[33] NICT. *World Record 402 Tb/s Transmission in a Standard Commercially Available Optical Fiber | 2024 | NICT - National Institute of Information and Communications Technology — nict.go.jp*. 2024. [Acessado em 04-10-2024]. Disponível em: <https://www.nict.go.jp/en/press/2024/06/26-1.html>.

[34] CHOU, D.; JIANG, M. A survey on data-driven network intrusion detection. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 54, n. 9, oct 2021. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3472753>.

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

[36] SCHLEGL, T. et al. f-anogan: Fast unsupervised anomaly detection with generative adversarial networks. *Medical Image Analysis*, v. 54, p. 30–44, 2019. ISSN 1361-8415. Disponível em: <https://doi.org/10.1016/j.media.2019.01.010>.

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

[38] ZENATI, H. et al. *Efficient GAN-Based Anomaly Detection*. 2019. Disponível em: <https://doi.org/10.48550/arXiv.1802.06222>.

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

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

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

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

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

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

[45] 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>.

[46] ALPAYDIN, E. *Introduction to Machine Learning*. 2nd. ed. [S.l.]: The MIT Press, 2010. ISBN 026201243X.

[47] FU, J. et al. Ganad: A gan-based method for network anomaly detection. *World Wide Web*, v. 26, n. 5, p. 2727–2748, Sep 2023. ISSN 1573-1413. Disponível em: <https://doi.org/10.1007/s11280-023-01160-4>.

[48] MOHAMMED, M.; KHAN, M. B.; BASHIER, E. B. M. *Machine learning: algorithms and applications*. [S.l.]: Crc Press, 2016. <https://doi.org/10.1201/9781315371658>. ISBN 1498705383.

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

[50] NIELSEN, M. *Neural Networks and Deep Learning*. Determination Press, 2015. Disponível em: <https://books.google.com.br/books?id=STDBswEACAAJ>.

[51] GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. ISBN 9780262035613.

[52] BASHAR, M. A.; NAYAK, R. Tanogan: Time series anomaly detection with generative adversarial networks. In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2020. Disponível em: <http://dx.doi.org/10.1109/SSCI47803.2020.9308512>.

[53] GERS, F.; SCHMIDHUBER, J.; CUMMINS, F. Learning to forget: continual prediction with lstm. In: *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*. [s.n.], 1999. v. 2, p. 850–855 vol.2. Disponível em: <https://doi.org/10.1049/cp:19991218>.

[54] KUREMOTO, T. et al. Forecast chaotic time series data by dbns. In: *2014 7th International Congress on Image and Signal Processing*. [s.n.], 2014. p. 1130–1135. Disponível em: <https://doi.org/10.1109/CISP.2014.7003950>.

[55] SUN, H. et al. Mts-dvgan: Anomaly detection in cyber-physical systems using a dual variational generative adversarial network. *Computers Security*, v. 139, p. 103570, 2024. ISSN 0167-4048. Disponível em: <https://doi.org/10.1016/j.cose.2023.103570>.

[56] SNOEK, J.; LAROCHELLE, H.; ADAMS, R. P. Practical bayesian optimization of machine learning algorithms. In: PEREIRA, F. et al. (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2012. v. 25. ISBN 9780262561457. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf>.

[57] WU, J. et al. Hyperparameter optimization for machine learning models based on bayesian optimizationb. *Journal of Electronic Science and Technology*, v. 17, n. 1, p. 26–40, 2019. ISSN 1674-862X. Disponível em: <https://doi.org/10.11989/JEST.1674-862X.80904120>.

[58] ARAUJO-FILHO, P. F. de et al. Unsupervised gan-based intrusion detection system using temporal convolutional networks and self-attention. *IEEE Transactions on Network and Service Management*, v. 20, n. 4, p. 4951–4963, 2023. Disponível em: <https://doi.org/10.1109/TNSM.2023.3260039>.

[59] LI, D. et al. Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks. In: TETKO, I. V. et al. (Ed.). *Artificial Neural Networks and Machine Learning – ICANN 2019: Text and Time Series*. Cham: Springer International Publishing, 2019. p. 703–716. ISBN 978-3-030-30490-4. Disponível em: <https://doi.org/10.48550/arXiv.1901.04997>.

[60] CHAKRABORTY, T. et al. *Ten Years of Generative Adversarial Nets (GANs): A survey of the state-of-the-art*. 2023. Disponível em: <https://doi.org/10.48550/arXiv.2308.16316>.

[61] PETERSON, M. *An Introduction to Decision Theory*. 2. ed. [S.l.]: Cambridge University Press, 2017. (Cambridge Introductions to Philosophy). ISBN 9781316585061.

[62] DASH, A.; YE, J.; WANG, G. A review of generative adversarial networks (gans) and its applications in a wide variety of disciplines: From medical to remote sensing. *IEEE Access*, v. 12, p. 18330–18357, 2021. Disponível em: <https://doi.org/10.1109/ACCESS.2023.3346273>.

[63] BROPHY, E. et al. Generative adversarial networks in time series: A systematic literature review. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 55, n. 10, fev. 2023. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3559540>.

[64] GULRAJANI, I. et al. *Improved Training of Wasserstein GANs*. 2017. Disponível em: <https://doi.org/10.48550/arXiv.1704.00028>.

[65] ARJOVSKY, M.; CHINTALA, S.; BOTTOU, L. *Wasserstein GAN*. 2017. Disponível em: <https://doi.org/10.48550/arXiv.1701.07875>.

[66] WAHEED, A. et al. Covidgan: Data augmentation using auxiliary classifier gan for improved covid-19 detection. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 8, p. 91916–91923, 2020. ISSN 2169-3536. Disponível em: <http://dx.doi.org/10.1109/ACCESS.2020.2994762>.

[67] KARRAS, T.; LAINE, S.; AILA, T. A style-based generator architecture for generative adversarial networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 43, n. 12, p. 4217–4228, 2021. Disponível em: <https://doi.org/10.1109/TPAMI.2020.2970919>.

[68] ZHANG, H. et al. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, p. 5908–5916, 2016. Disponível em: <https://doi.org/10.48550/arXiv.1612.03242>.

[69] RADFORD, A.; NARASIMHAN, K. *Improving Language Understanding by Generative Pre-Training*. 2018. Disponível em: <https://api.semanticscholar.org/CorpusID:49313245>.

[70] RADFORD, A. et al. *Language Models are Unsupervised Multitask Learners*. 2019. Disponível em: <https://api.semanticscholar.org/CorpusID:160025533>.

[71] BROWN, T. B. et al. *Language Models are Few-Shot Learners*. 2020. Disponível em: <https://doi.org/10.48550/arXiv.2005.14165>.

[72] OPENAI et al. *GPT-4 Technical Report*. 2024. Disponível em: <https://doi.org/10.48550/arXiv.2303.08774>.

[73] BAHDANAU, D.; CHO, K.; BENGIO, Y. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016. Disponível em: <https://doi.org/10.48550/arXiv.1409.0473>.

[74] LONGO, L. et al. Explainable artificial intelligence (xai) 2.0: A manifesto of open challenges and interdisciplinary research directions. *Information Fusion*, v. 106, p. 102301, 2024. ISSN 1566-2535. Disponível em: <https://doi.org/10.1016/j.inffus.2024.102301>.

[75] NEUPANE, S. et al. Explainable intrusion detection systems (x-ids): A survey of current methods, challenges, and opportunities. *IEEE Access*, v. 10, p. 112392–112415, 2022. Disponível em: <https://doi.org/10.1109/ACCESS.2022.3216617>.

[76] SAEED, W.; OMLIN, C. Explainable ai (xai): A systematic meta-survey of current challenges and future opportunities. *Knowledge-Based Systems*, v. 263, p. 110273, 2023. ISSN 0950-7051. Disponível em: <https://doi.org/10.1016/j.knosys.2023.110273>.

[77] GASPAR, D.; SILVA, P.; SILVA, C. Explainable ai for intrusion detection systems: Lime and shap applicability on multi-layer perceptron. *IEEE Access*, v. 12, p. 30164–30175, 2024. Disponível em: <https://doi.org/10.1109/ACCESS.2024.3368377>.

[78] HAQUE, A. B.; ISLAM, A. N.; MIKALEF, P. Explainable artificial intelligence (xai) from a user perspective: A synthesis of prior literature and problematizing avenues for future research. *Technological Forecasting and Social Change*, v. 186, p. 122120, 2023. ISSN 0040-1625. Disponível em: <https://doi.org/10.1016/j.techfore.2022.122120>.

[79] RJOUB, G. et al. A survey on explainable artificial intelligence for cybersecurity. *IEEE Transactions on Network and Service Management*, v. 20, n. 4, p. 5115–5140, 2023. Disponível em: <https://doi.org/10.1109/TNSM.2023.3282740>.

[80] PAWLICKI, M. et al. Advanced insights through systematic analysis: Mapping future research directions and opportunities for xai in deep learning and artificial intelligence used in cybersecurity. *Neurocomputing*, v. 590, p. 127759, 2024. ISSN 0925-2312. Disponível em: <https://doi.org/10.1016/j.neucom.2024.127759>.

[81] BRASILEIRO, G. *Lei Geral de Proteção de Dados Pessoais (LGPD)*. 2024. Acesso em 02-12-2024. Disponível em: <https://www.gov.br/esporte/pt-br/acesso-a-informacao/lgpd>.

[82] CAPUANO, N. et al. Explainable artificial intelligence in cybersecurity: A survey. *IEEE Access*, v. 10, p. 93575–93600, 2022. Disponível em: <https://doi.org/10.1109/ACCESS.2022.3204171>.

[83] PHILLIPS, P. J. et al. *Four Principles of Explainable Artificial Intelligence*. NIST Interagency/Internal Report (NISTIR), National Institute of Standards and Technology, Gaithersburg, MD, 2021. Disponível em: <https://doi.org/10.6028/NIST.IR.8312>.

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

[85] LUNDBERG, S.; LEE, S.-I. *A Unified Approach to Interpreting Model Predictions*. 2017. Disponível em: <https://doi.org/10.48550/arXiv.1705.07874>.

[86] ROTH, A. E. (Ed.). *The Shapley Value: Essays in Honor of Lloyd S. Shapley*. [S.l.]: Cambridge University Press, 1988. <https://doi.org/10.1017/CBO9780511528446>. ISBN 9780511528446.

[87] APLEY, D. W.; ZHU, J. Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, v. 82, n. 4, p. 1059–1086, 06 2020. ISSN 1369-7412. Disponível em: <https://doi.org/10.1111/rssb.12377>.

[88] RIBEIRO, M. T.; SINGH, S.; GUESTRIN, C. *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*. 2016. Disponível em: <https://doi.org/10.48550/arXiv.1602.04938>.

[89] M., G.; SETHURAMAN, S. C. A comprehensive survey on deep learning based malware detection techniques. *Computer Science Review*, v. 47, p. 100529, 2023. ISSN 1574-0137. Disponível em: <https://doi.org/10.1016/j.cosrev.2022.100529>.

[90] SHARAFALDIN, I. et al. Developing realistic distributed denial of service (ddos) attack dataset and taxonomy. In: *2019 International Carnahan Conference on Security Technology (ICCST)*. [s.n.], 2019. p. 1–8. Disponível em: <https://doi.org/10.1109/CCST.2019.8888419>.

[91] GROUP, O. R. *Datasets used in Publications*. 2024. Acesso em 02-12-2024. Disponível em: <http://www.uel.br/grupos/orion/datasets.html>.

[92] MOUSTAFA, N.; SLAY, J. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In: *2015 Military Communications and Information Systems Conference (MilCIS)*. [s.n.], 2015. p. 1–6. Disponível em: <https://doi.org/10.1109/MilCIS.2015.7348942>.

[93] JIANG, Y.; CHANG, S.; WANG, Z. *TransGAN: Two Pure Transformers Can Make One Strong GAN, and That Can Scale Up*. 2021. Disponível em: <https://doi.org/10.48550/arXiv.2102.07074>.

[94] LI, Y. et al. Dct-gan: Dilated convolutional transformer-based gan for time series anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, v. 35, n. 4, p. 3632–3644, 2023. Disponível em: <https://doi.org/10.1109/TKDE.2021.3130234>.

[95] GUMMADI, A. N.; NAPIER, J. C.; ABDALLAH, M. Xai-iot: An explainable ai framework for enhancing anomaly detection in iot systems. *IEEE Access*, v. 12, p. 71024–71054, 2024. Disponível em: <https://doi.org/10.1109/ACCESS.2024.3402446>.

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

[97] PEDREGOSA, F. et al. *Scikit-learn: Machine Learning in Python*. 2018. Disponível em: <https://doi.org/10.48550/arXiv.1201.0490>.

[98] DODGE, Y. *The Concise Encyclopedia of Statistics*. Springer New York, 2008. (The Concise Encyclopedia of Statistics). ISBN 9780387317427. Disponível em: <https://books.google.com.br/books?id=k2zklGOBRDwC>.

[99] BOPPANA, T. K.; BAGADE, P. Gan-ae: An unsupervised intrusion detection system for mqtt networks. *Engineering Applications of Artificial Intelligence*, v. 119, p. 105805, 2023. ISSN 0952-1976. Disponível em: <https://doi.org/10.1016/j.engappai.2022.105805>.

[100] NOGUEIRA, F. *Bayesian Optimization: Open source constrained global optimization tool for Python*. 2014–. Acessado em 04-12-2024. Disponível em: <https://github.com/bayesian-optimization/BayesianOptimization>.

[101] KLAISE, J. et al. Alibi explain: Algorithms for explaining machine learning models. *Journal of Machine Learning Research*, v. 22, n. 181, p. 1–7, 2021. Disponível em: <http://jmlr.org/papers/v22/21-0017.html>.

TRABALHOS PUBLICADOS PELO AUTOR

Trabalhos publicados pelo autor durante o programa.

Publicações principais do trabalho.

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 Jr., **Anomaly and intrusion detection using deep learning for software-defined networks: A survey**, Expert Systems with Applications, dezembro/2024, Elsevier, Volume 256, ISSN: 0957-4174, DOI: 10.1016/j.eswa.2024.124982. (Qualis CC 2017, A1)