Universidade Estadual de Londrina

DANIEL MATHEUS BRANDÃO LENT

DETECÇÃO DE ANOMALIAS UTILIZANDO REDES NEURAIS PROFUNDAS RECORRENTES E LÓGICA FUZZY

LONDRINA
2023

DANIEL MATHEUS BRANDÃO LENT

DETECÇÃO DE ANOMALIAS UTILIZANDO REDES NEURAIS PROFUNDAS RECORRENTES E LÓGICA FUZZY

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

Orientador: Prof. Dr. Mario Lemes Proença Jr.

LONDRINA

2023

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

B817d Lent, Daniel Matheus Brandão.
Detecção de anomalias utilizando redes neurais profundas recorrentes e lógica fuzzy / Daniel Matheus Brandão Lent. - Londrina, 2023.
68 f. : il.

Orientador: Mario Lemes Proença Jr..
Dissertação (Mestrado em Ciência da Computação) - Universidade Estadual de Londrina, Centro de Ciências Exatas, Programa de Pós-Graduação em Ciência da Computação, 2023.
Inclui bibliografia.

1. Segurança de Redes - Tese. 2. Redes Neurais Artificiais - Tese. 3. Aprendizagem Profunda - Tese. 4. Detecção de Anomalias - Tese. I. Lemes Proença Jr., Mario. II. Universidade Estadual de Londrina. Centro de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU 519

DANIEL MATHEUS BRANDÃO LENT

DETECÇÃO DE ANOMALIAS UTILIZANDO REDES NEURAIS PROFUNDAS RECORRENTES E LÓGICA FUZZY

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

BANCA EXAMINADORA

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

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

Prof. Dr. José Valdeni de Lima
Instituto de Informática da Universidade Federal do Rio Grande do Sul – UFRGS

Londrina, 14 de Fevereiro de 2023.

Este trabalho é dedicado aos meus pais, que sempre valorizaram minha educação.

AGRADECIMENTOS

Agradeço e dedico esse trabalho aos meus pais que sempre me incentivaram a estudar e possibilitaram toda a minha formação. Esse trabalho não seria possível sem o apoio que recebi. Também agradeço à minha família que, mesmo com pouco contato, me estimulou a estudar e me aprimorar.

Agradeço aos amigos que fiz em Londrina, especialmente Henrique, Matheus, Vinícius, e Vitor. Aprendi e continuo aprendendo muito com eles ao mesmo tempo que recebi suporte em momentos difíceis. Espero enriquecer suas vidas da mesma forma que fizeram com a minha.

Agradeço os membros do grupo Orion de pesquisa, especialmente ao professor Mario Lemes Proença Jr. pelo convite que deu origem a esse trabalho e toda a sua paciência enquanto trabalhamos juntos. Sua orientação eleva o nível de qualquer trabalho e me faz evoluir como pesquisador. Agradeço também ao doutorando Matheus Novaes por me ajudar em diversos momentos durante a pesquisa. Agradeço ao mais novo membro do grupo, Vitor, que já colabora com a pesquisa.

Por fim, agradeço à Universidade Estadual de Londrina por proporcionar o curso de mestrado em ciência da computação e que acolhe o trabalho do grupo de pesquisa. Agradeço ao departamento de computação por colaborar com o financiamento do artigo originado deste trabalho. Agradeço à fundação Araucária pelo financiamento do trabalho.

"O temor do Senhor é instrução da sabedoria, e a humildade precede a honra."
(Bíblia Sagrada, Provérbios 15:33)

LENT, D. M. B.. **Detecção de Anomalias Utilizando Redes Neurais Profundas Recorrentes e Lógica Fuzzy**. 2023. 68f. Dissertação (Mestrado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2023.

RESUMO

A Internet é essencial para a sociedade atual e está presente continuamente na vida das pessoas. Compras *online*, serviços de armazenamento de dados, mapas e outros serviços só são possíveis com uma conexão com a rede. Por serem tão indispensáveis, os avanços tecnológicos de dispositivos que mantém servidores funcionando e conectados à Internet são constantes. Paradigmas como as redes definidas por *software* são implementados para simplificar a manutenção e gerenciamento da rede. Para isso, um controlador centralizado é utilizado para gerenciar toda a rede, o que facilita a instalação e configuração de dispositivos. Contudo, esse controlador é um alvo para ataques distribuídos de negação de serviço, que podem interromper seu funcionamento por tempo indeterminado. Nesse trabalho, um sistema de detecção e mitigação de anomalias utilizando redes neurais recorrentes profundas e lógica difusa é proposto. O sistema utiliza as redes neurais para prever o comportamento do tráfego e lógica difusa para comparar a previsão com os valores reais. Ao detectar uma discrepância entre resultados, o módulo de mitigação é ativado e o tráfego malicioso é bloqueado. O sistema foi validado utilizando dados do grupo Orion de Redes de Computadores do departamento de computação da Universidade Estadual de Londrina e o conjunto de dados CICDDoS2019 da universidade de Nova Brunswick.

Palavras-chave: Aprendizagem de Máquina; Detecção de Anomalia; Redes Definidas por *Software*; *Gated Recurrent Unit*; Lógica *Fuzzy*.

LENT, D. M. B.. **Anomaly Detection Using Deep Recurrent Neural Networks and Fuzzy Logic**. 2023. 68p. Master's Thesis (Master in Science in Computer Science) – State University of Londrina, Londrina, 2023.

ABSTRACT

The Internet is essential for society and is continuously present in people's life. Online shopping, cloud data storage, maps, and other services are only possible due to the web. For being so necessary, technological advancements of devices that keep servers running and connected to the Internet are incessant. Paradigms such as Software Defined Networking are implemented to simplify network's maintance and management. To do so, a centralized controller is used to commmand the whole network, which facilitates setup and administration. However, this controller is susceptible to ditributed denial of service attacks, that may terminate the network's operations indefinetly. In this work, an anomaly detection and mitigation system using recurrent neural networks and fuzzy logic is proposed. This method used neural networks to forecast traffic and fuzzy logic to compare the prediction with real values. When a discrepancy is detected, the mitigation module is activated and malicious traffic is blocked. The system was tested using data from the data communication and networking research group called Orion, from computer science department at State University of Londrina called Orion.

Keywords: Machine Learning; Anomaly Detection; Software Defined Networks; Gated Recurrent Unit; Fuzzy Logic

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

5G	*5th Generation Mobile Network*
6G	*6th Generation Mobile Network*
AUC	*Area Under Curve*
CNN	*Convolutional Neural Network*
DDoS	*Distributed Denial of Service*
DNS	*Domain Name System*
DoS	*Denial of Service*
GAN	*Generative Adversarial Network*
GB	*Gigabyte*
GRU	*Gated Recurrent Unit*
IDS	*Intrusion Detectio System*
IoT	*Internet of Things*
IP	*Internet Protocol*
LSTM	*Long Short-Term Memory*
RAM	*Random Access Memory*
ReLU	*Rectified Linear Unit*
ROC	*Receiver operating characteristic*
SDN	*Software Defined Networks*
TCN	*Temporal Convolutional Networks*
UEL	*Universidade Estadual de Londrina*
Wi-Fi	*Wireless Fidelity*

LISTA DE SÍMBOLOS

Σ	Letra grega Sigma, maiúscula, indica somatório
σ	Letra grega sigma, minúscula, se refere à função de ativação *sigmoid*
$tanh()$	Função tangente hiperbólica
h	Horas
H	Entropia
p_i	Probabilidade de i
e	Número de Euler
x	Dados de entrada
W	Primeiro conjunto de pesos de uma rede neural GRU
V	Segundo conjunto de pesos de uma rede neural GRU
b	Terceiro conjunto de pesos de uma rede neural GRU
r_i	Valor de *reset* de uma rede neural GRU
z	Valor do portão de *update* de uma rede neural GRU
h_t	Estado oculto da rede neural GRU

SUMÁRIO

1 INTRODUÇÃO

A Internet é essencial para a sociedade atual. Ela está presente constantemente na vida das pessoas por meio de diversos dispositivos, principalmente computadores e celulares. Por ser versátil, a rede é utilizada em diferentes tipos de aplicações, como comunicação, compras *online*, serviços de armazenamento de dados e outros. Mesmo sendo tão predominante, cada vez mais novas formas de utilizar a Internet são criadas e implementadas. Um exemplo disso são as cidades inteligentes, que propõem utilizar dados coletados por diferentes tipos de sensores para otimizar políticas públicas [1][2]. Essas e outras aplicações se tornam ainda mais viáveis com a evolução de tecnologias de comunicação, como o 5G e o *Wi-Fi* 6, com maiores taxas de transferência e baixa latência [3][4]. Por isso, espera-se que a dependência da sociedade pela Internet aumente ainda mais ao longo dos anos.

Por serem tão indispensáveis, essas aplicações precisam manter seus servidores funcionando continuamente para manter sua disponibilidade. Para isso, é preciso que uma infraestrutura para gerenciamento seja regularmente configurada, monitorada e gerenciada. Alguns fatores podem tornar esse processo mais complexo, como a adição de novos equipamentos, configuração de hardware de múltiplos fabricantes, diferentes sistemas operacionais, dentre outros. Essa complexidade elevada pode inviabilizar o crescimento de uma rede, o que limita a adição de melhorias e novas aplicações. Nesse contexto, o paradigma de redes definidas por *software* (SDN) foi criado e se tornou uma possível solução. Com ele, um controlador centralizado fica responsável por enviar instruções de encaminhamento para cada *switch* na rede utilizando um único protocolo. Além disso, esse controlador é programável, e opera como uma interface para monitorar e controlar os ativos de rede. Dessa forma, por meio do controlador, é possível implementar sistemas de alocação dinâmica de tráfego [5] e sistemas de segurança que operam de forma escalável [6].

Um dos maiores problemas de uma rede SDN é que o controlador se torna um alvo para ataques distribuídos de negação de serviço (DDoS) [7][8]. Esse tipo de ataque tem como objetivo esgotar os recursos de seu alvo ao lhe enviar requisições de forma que seu serviço fique indisponível para outros usuários por tempo indefinido. Em uma rede SDN, esse tipo de ataque é ainda mais danoso, pois pode sobrecarregar o controlador, o que interromperia o bom funcionamento de toda a rede [9][10]. Para mitigar o ataque, seria necessário bloquear o tráfego da maioria dos dispositivos maliciosos. Contudo, isso não é uma tarefa trivial, já que os pacotes maléficos estão misturados com os legítimos e têm origens variadas [11]. Para resolver esse problema, é fundamental um sistema dedicado de detecção e mitigação de anomalias.

Para identificar ataques, sistemas de detecção de intrusão baseados em anomalia utilizam dados de tráfego coletados ao longo do tempo. A partir dos dados, uma definição de normalidade também conhecida como *baseline* é traçada. Ao definir o que é normal, é possível indicar como anomalia todo o comportamento que divirja o suficiente dessa caracterização baseado em algum limiar de decisão [12]. Esse tipo de detecção tem como vantagem a possibilidade de apontar anomalias desconhecidas, já que não precisa de uma assinatura apresentada previamente [13]. Entretanto, é importante que sua previsão se mantenha atualizada e precisa em relação ao tráfego real.

Uma das formas de implementar tal caracterização é utilizando redes neurais profundas. Redes neurais se mostraram superiores a outros métodos de aprendizagem de máquina para detecção de anomalias devido à sua capacidade de generalização e à possibilidade de aproveitar grandes quantidades de dados mais eficientemente [14][15]. Elas frequentemente têm seus neurônios adaptados para cada tipo de problema, o que pode melhorar sua performance ainda mais. Nesse caso, os dados de tráfego podem ser considerados parte de uma série temporal, por isso redes neurais recorrentes tendem a ser utilizadas para detectar comportamentos anômalos [16][17]. Esse tipo de rede possui neurônios especiais, que podem receber valores de saídas anteriores, ou até armazenar temporariamente informações de contexto, para influenciar as próximas decisões [18]. Isso viabiliza que anomalias que requeiram contexto sejam detectadas, melhorando seu desempenho [17].

O presente estudo propõe a utilização de redes neurais recorrentes para traçar o comportamento normal do tráfego e fazer previsões sobre a movimentação dos próximos segundos. Em seguida, lógica difusa [19] será aplicada para comparar a previsão com o tráfego real, apontando anomalias quando o erro for maior que determinado limiar. Quando uma anomalia é detectada, o módulo de mitigação do sistema proposto é acionado para identificar e bloquear os endereços atacantes. Para validação, modelo foi testado em duas bases de dados: uma delas produzida pelo grupo Orion de Redes de Computadores do Departamento de Computação da Universidade Estadual de Londrina e a outra, pela Universidade de Nova Brunswick, chamada CICDDoS2019.

As principais contribuições do trabalho são:

- O desenvolvimento de um sistema de detecção de anomalias que combina redes neurais recorrentes e lógica difusa, com o objetivo de detectar ataques de DDoS e *portscan* em no máximo 1 segundo;
- Um algoritmo que identifica e bloqueia atacantes de DDoS e *portscan* antes que o controlador SDN seja afetado pelo ataque;
- O modelo proposto é comparado a outros tipos de redes neurais profundas: tradicionais, convolucionais, temporais convolucionais e *long short-term memory*, para

avaliar sua eficiência;

- O modelo foi testado com duas bases de dados: uma desenvolvida pelo grupo de pesquisa Orion do Departamento de Computação da UEL e outra desenvolvida pela Universidade de Nova Brunswick que tem sido utilizada pela comunidade científica [20][21][22].

O restante desse trabalho está dividido da seguinte forma: o capítulo 2 apresenta a fundamentação teórica e o capítulo 3, os trabalhos relacionados. Os capítulos 4 e 5 mostram o sistema proposto e o estudo de caso, respectivamente. Por fim, o capítulo 6 apresenta a conclusão e os próximos passos desse estudo.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Redes Definidas por *Software*

Redes definidas por *software* (SDN, do inglês *Software Defined Network*) existem como um paradigma alternativo às redes tradicionais. Sua aplicação permite que novas e modernas estruturas de rede sejam construídas, além de possibilitar a evolução de novos paradigmas como 5G, 6G, computação em nuvem, sistemas ciber-físicos e Internet das Coisas [23][24]. Fato é que redes SDN são superiores às tradicionais por solucionar alguns de seus problemas críticos. Redes tradicionais tendem a ser heterogêneas, com equipamentos de diferentes fabricantes que possuem *software* com incompatibilidade entre si. Em redes tradicionais, cada dispositivo precisa ser configurado individualmente, o que gera problemas de gerenciamento e escalabilidade durante a criação de novas políticas e adição de novas funções [25].

Para resolver tais problemas, redes definidas por *software* centralizam o plano de controle em um controlador, que funciona como interface para o gerente de rede e quaisquer programas de gerenciamento e automação a serem instalados. A partir do controlador, cada *switch* pode ser configurado de forma independente, inclusive possibilitando automações e distribuição de recursos sob demanda, aumentando a eficiência da rede e a qualidade de serviço [26]. As instruções de encaminhamento vindas do plano de controle são armazenadas em cada *switch* em uma tabela. Essa tabela é atualizada sempre que uma entrada não compatível é encontrada ou por meio de novas instruções do controlador, possibilitando um roteamento resiliente e adaptável [27].

Paralelamente ao plano de controle, o plano de dados é encarregado de transmitir pacotes de acordo com as regras de encaminhamento. Como já explicado, as regras vêm do plano de controle e de qualquer tipo de *software* que utilize o controlador para gerenciar a rede por meio da interface *southbound*, que faz a comunicação de dispositivos com o controlador. As tabelas de encaminhamento previamente definidas são importantes devido ao possível volume de tráfego que pode ocorrer. Sem sua existência, seria necessário enviar requisições ao controlador para cada novo pacote na rede, o que sobrecarregaria o sistema rapidamente. A figura 1 mostra como uma rede SDN é normalmente organizada.

A utilização desse paradigma cria novas questões sobre segurança de rede. Uma delas é a importância do controlador e como ele pode ser um alvo prioritário de ataques, visto que seu comprometimento prejudicaria o funcionamento da rede como um todo. Por isso, um assunto estudado extensivamente é a segurança de SDN [28][29][30]. Alguns recursos proporcionados pela SDN são chave para implementação de medidas de segurança. Dentre eles estão a visibilidade da rede como um todo com a possibilidade de

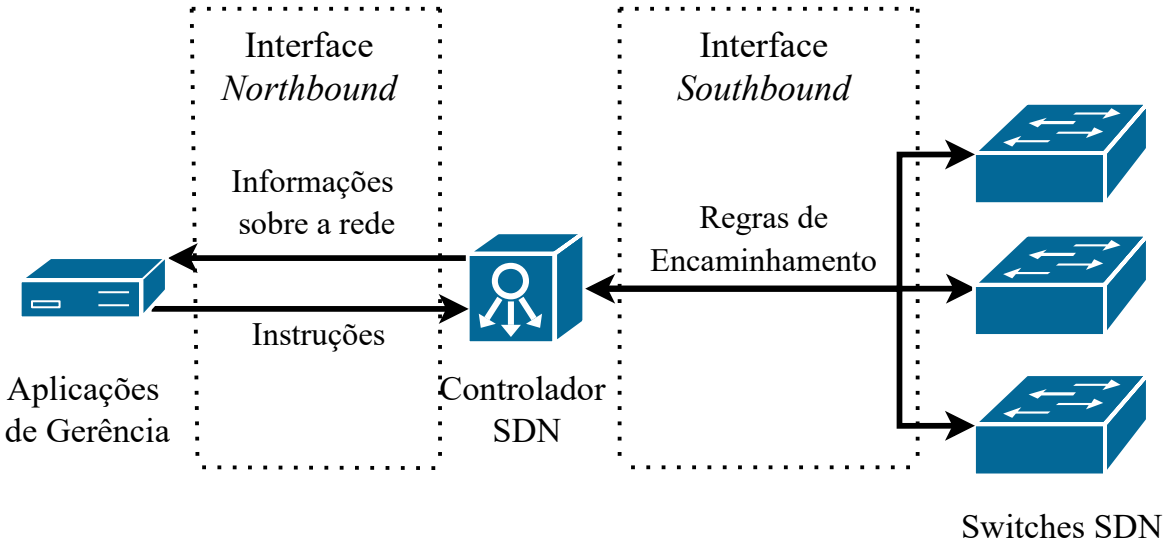

Figura 1 – Representação de uma Rede SDN

controle dinâmico de fluxo, além da capacidade de reprogramar a rede, alterando funções de dispositivos conforme a necessidade [31] [32].

2.2 Detecção de Anomalias e Sistemas de Detecção de Intrusão

Métodos de detecção de anomalias são estudados para serem aplicados em diversas áreas, como reconhecimento de imagens, manutenção de maquinário industrial, e monitoramento de tráfego de rede [33][34][35]. Para redes de computadores, os sistemas de detecção de intrusão (IDS, do inglês *Intrusion Detection System*) são essenciais para manter servidores livres de ameaças, como ataques de negação de serviço, que podem facilmente atrapalhar a operação do sistema e causar graves prejuízos. Um IDS é uma combinação de *software* e *hardware* que monitora a rede e detecta ações maliciosas. Existem diferentes formas de classificar sistemas de detecção de intrusão. É possível classificar pelo método que fazem a detecção, sendo chamados de "*signature-based*" e "*anomaly-based*", ou pela plataforma que coleta os dados a serem analisados, sendo chamados de "*Network-Based*", "*Host-Based*", e Híbrido [36].

Sistemas *Host Based* normalmente funcionam dentro do host que está sendo monitorado e têm como objetivo a proteção de apenas um computador. Por se tratar de um único dispositivo, técnicas de monitoramento específicas podem ser utilizadas para detectar não só ataques contra o *host*, mas também ações maliciosas internas, como abuso de privilégios e acesso indevido de recursos.

Por outro lado, os sistemas *Network Based* monitoram diversos dispositivos conectados em uma rede. O monitoramento é feito utilizando dados de tráfego de cada *host*,

analisando dados como endereço *IP* (*Internet Protocol*), porta e até conteúdo de pacotes, em alguns casos. Uma das dificuldades desse tipo de sistema está na quantidade de tráfego que deve ser analisado. Para redes em maior escala pode ser inviável monitorar tantas informações em tempo real, o que obriga esse tipo de sistema a utilizar menos informações de cada pacote ou mudar a forma como o tráfego é representado para o sistema.

Sistemas híbridos são uma combinação dos anteriores, utilizando dados coletados por hosts juntamente com dados de toda rede. Esse tipo de sistema possibilita que diferentes tipos de ameaças sejam detectados, mas é consideravelmente mais difícil de se implementar, por exigir a combinação de técnicas de análise de tráfego com métodos de detecção de intrusão.

Sistemas *signature-based* utilizam padrões aprendidos de ataques para detectar ameaças conhecidas. Esse tipo de sistema tem como vantagem uma alta precisão para detectar ataques conhecidos e um pouco de suas variações, ao mesmo tempo em que apresenta uma baixa taxa de alarmes falsos. A desvantagem desse tipo de método é que dificilmente um ataque nunca antes visto será detectado, já que sua assinatura não será reconhecida [37].

Em contrapartida, sistemas *anomaly-based* são desenvolvidos para reconhecer o comportamento normal de uma rede enquanto apontam qualquer desvio como anomalia. Esse tipo de sistema pode detectar anomalias nunca antes vistas, já que não atua a partir do reconhecimento da assinatura do ataque. Contudo, é comum que alarmes falsos sejam mais frequentes do que aqueles baseados em assinatura, já que anomalias que não são ataques podem ocorrer e mudar o perfil de tráfego. Esse tipo de sistema também pode sofrer problemas de deriva de conceito, já que o perfil da rede pode se alterar com o tempo, criando a necessidade de atualizar a definição de "normal" do sistema ao longo do tempo. Por fim, identificar ataques não é trivial para esse tipo de sistema sem utilizar algum tipo de assinatura pós detecção [38].

Neste trabalho, será apresentado um sistema *network-based* e *anomaly-based*. A aplicação de uma SDN é fundamental para promover a "visão" do tráfego de toda a rede por um único *software*, que monitora e envia instruções de mitigação a medida que detecta comportamento que divirja do normal aprendido. Maior detalhamento do sistema será evidenciado no capítulo 4.

2.2.1 Ataques de *Portscan*

No presente trabalho, dois tipos principais de anomalias serão abordados: os ataques de portscan e os ataques distribuídos de negação de serviço. Um ataque de portscan é uma forma de coletar informações, como um ataque de reconhecimento [39]. Com a informação em mãos, o atacante pode planejar um próximo ataque explorando alguma vulnerabilidade que possa ter sido encontrada.

Para coletar as informações, o atacante envia pacotes para diferentes portas da vítima e usa as respostas como referência se há algum serviço habilitado naquela porta ou se está fechada. Esse tipo de ataque pode vir na forma mas agressiva, de força bruta que envia diversos pacotes de uma vez, ou numa forma discreta, em que poucos, ou apenas um pacote é enviado apenas para sinalizar o início de uma ligação TCP, mas nunca finaliza o processo, já que basta saber que uma conexão seria formada [40].

2.2.2 Ataques de Negação de Serviço

Ataques de negação de serviço (DoS, do inglês *Denial of Service*) já são conhecidos há décadas por seu potencial de inutilizar aplicações pela Internet. Tradicionalmente, esse tipo de ataque acontece com um atacante abrindo requisições de um serviço sem prosseguilas, até que o servidor tenha seus recursos esgotados e não possa abrir mais requisições, impossibilitando que usuários legítimos tenham acesso [41].

Ataques DoS são simples de serem neutralizados, basta reconhecer e bloquear o endereço remetente da maior quantidade de requisições. Contudo, esses ataques começaram a ser realizados de forma distribuída (DDoS, do inglês *Distributed Denial of Service*), o que significa que diversos computadores atacantes agem simultaneamente para derrubar um serviço. Essa modalidade é mais grave por dois motivos principais: o maior número de computadores atacantes gera um volume de requisições significativamente maior do que o tradicional; outro fator relevante é que o ataque não é simples de mitigar, pois não é trivial diferenciar usuários legítimos de atacantes.

2.3 Redes Neurais

Redes neurais têm sido utilizadas há anos como método de aprendizagem de máquina. Inspiradas no cérebro humano, redes neurais utilizam neurônios como estruturas adaptáveis que se organizam em camadas por meio de conexões sinápticas. O aprendizado se dá no ajuste dessas conexões durante o treinamento. Um neurônio é uma estrutura simples que possui um conjunto de pesos que são combinados com uma entrada e formam uma saída. Normalmente os neurônios também possuem uma função de ativação, que determina a imagem de sua saída e evita a linearidade da rede neural. Uma camada é um conjunto de neurônios organizados em paralelo em que todos os neurônios recebem a mesma entrada. As camadas são organizadas em série e formam a rede neural. Dessa forma, a informação sai da camada de entrada e percorre a rede neural, sendo alterada conforme os pesos e funções de ativação são aplicados. A equação 2.1 descreve o funcionamento de um perceptron, em que f é a função de ativação do neurônio, b é um dos pesos, n representa a quantidade de elementos na entrada, x_i, os elementos da entrada e w_i, o restante dos pesos.

$$y = f(b + \sum_{i=0}^{n} x_i w_i) \tag{2.1}$$

Redes neurais podem ser utilizadas para criar principalmente dois tipos de modelos de aprendizagem. O primeiro é um modelo de classificação, em que a função da rede neural consiste em indicar, dentre um conjunto finito de opções, uma classe para determinada entrada, conforme os padrões aprendidos. O aprendizado ocorre com diversos exemplos de cada classe sendo apresentados ao modelo até que a rede neural consiga diferenciá-los. O segundo tipo de modelo é o de regressão, em que o objetivo é relacionar um conjunto de variáveis com um valor numérico. Quando treinado, o sistema descreve uma função que encontra valores para entradas desconhecidas. No sistema utilizado neste trabalho, redes neurais foram treinadas para "aprender" o comportamento normal da rede de computadores e, então, prever esse comportamento como uma regressão.

Durante a fase de execução, após o treinamento de uma rede neural, o ajuste de pesos é interrompido e a rede será utilizada apenas para responder sobre as entradas a ela apresentadas. Se a rede neural conseguir bons resultados com o conjunto de treinamento e este representar o conjunto de testes com sucesso, espera-se que o desempenho do modelo seja satisfatório. Contudo, um dos defeitos desse tipo de modelo é que na maioria das vezes não é óbvio qual a lógica utilizada pela rede neural para alcançar uma resposta correta.

O problema de interpretabilidade de redes neurais se torna mais grave quanto maior a quantidade de camadas e neurônios, já que a quantidade de cálculos e parâmetros treináveis aumenta proporcionalmente. Apesar do modelo funcionar corretamente, é interessante para qualquer pesquisa conhecer a razão por trás de seus resultados. Por isso, modelos de redes neurais que explicam a influência de cada *feature* no resultado final têm sido desenvolvidos. Esse tipo de modelo utiliza uma variação nas entradas do modelo para criar explicações, sejam elas estatísticas, visuais ou de outro tipo [42].

2.3.1 Redes Neurais Profundas

Redes neurais profundas se tornaram mais populares conforme o poder de processamento de computadores aumentou. Esse tipo de rede é caracterizado por possuir mais camadas ocultas, o que a torna mais complexa, difícil de treinar e lenta. Contudo, modelos profundos têm uma capacidade de generalização maior do que redes rasas, conseguindo melhores resultados quando dados suficientes são providenciados [43]. Um dos problemas de redes neurais profundas é o chamado desaparecimento do gradiente, que torna o aprendizado das camadas iniciais mais lento conforme o número de camadas aumenta [44]. Esse problema é combatido com técnicas de treinamento e com a utilização de funções de ativação que reduzam seu efeito [45].

A partir da rede neural tradicional, novos tipos de redes podem ser criados ao alterar o funcionamento dos neurônios e a organização das camadas. Neurônios especiais são utilizados em redes como *Gated Recurrent Unit* (GRU), *Long short-term memory* (LSTM), e redes convolucionais (CNN, do inglês *Convolutional Neural Network*). Outros tipos de redes neurais podem também ser criados utilizando arquiteturas diferentes, como *autoencoders* e redes generativas adversárias.

2.3.2 Redes de *Gated Recurrent Unit*

A *gated recurrent unit* (GRU) - que pode ser traduzida como "unidades recorrentes com portões" - é um tipo de rede neural que utiliza um neurônio especial. Esse neurônio possui unidades de memória que armazenam um "estado" que influencia as próximas respostas da rede neural. Dessa forma, a rede neural consegue utilizar acontecimentos anteriores em decisões futuras, o que a torna mais eficiente ao se adaptar a séries temporais, sequências ou qualquer tipo de problema que envolva relações dinâmicas entre amostras [46][47][48]. Alguns exemplos de usos de redes GRU são: processamento de vídeos [49]; processamento de linguagem natural [50]; composição musical [51], dentre outros.

O funcionamento desse tipo de neurônio pode ser descrito em quatro equações [46]. Cada neurônio possui três conjuntos de pesos: W, que é sempre combinado com a entrada x_t do neurônio, V, que é sempre combinado com a saída anterior h_{t-1} do neurônio, e b, que é um valor independente chamado de viés. Esses valores são ajustados durante o treinamento e normalmente não são alterados durante a execução do sistema.

Primeiramente é calculado o valor de r_t, que representa o "*reset*". Esse valor define o quanto de informação da última saída será descartada para essa iteração. Seu cálculo é definido na equação 2.2. Em seguida, é calculado o valor de $\hbar_t$, que é o estado oculto depois de removida a informação do portão de "*reset*", representado na equação 2.3. Nessa equação, *tanh* é a função de tangente hiperbólica. A terceira equação, representada em 2.4, define o valor do portão de *update* z_t, que será utilizado para calcular o novo estado oculto ao ser combinado com o estado oculto anterior. Nessa equação, σ representa a função *sigmoid*. Por fim, a equação 2.5 calcula a saída do neurônio h_t, utilizando os valores das equações anteriores.

$$r_t = \sigma_g(W_r x_t + V_r h_{t-1} + b_r) \tag{2.2}$$

$$\hbar_t = tanh(W_h x_t + V_h(r_t \cdot h_{t-1}) + b_h) \tag{2.3}$$

$$z_t = \sigma(W_z x_t + V_z h_{t-1} + b_z) \tag{2.4}$$

$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \hbar_t \tag{2.5}$$

As funções *sigmoid* e tangente hiperbólica são definidas pelas seguintes equações respectivamente:

$$sigmoid(x) = \frac{1}{1 - e^{-x}} \tag{2.6}$$

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.7}$$

2.3.3 Redes Convolucionais

Redes Neurais Convolucionais (CNN) têm como principal característica o processo de convolução que alguns de seus neurônios possuem. As convoluções são úteis para extrair características importantes e também simplificar a computação dos dados, diminuindo sua complexidade. Essa simplificação é essencial para esse tipo de rede, pois possibilita sua utilização no processamento de imagens; processo que tende a ser mais complexo que outros por considerar cada *pixel* como uma *feature* [52]. Como exemplo, o processamento de uma imagem de $720x720$ *pixels* por uma rede neural tradicional implicaria na utilização de $518,400$ pesos para cada neurônio da primeira camada, enquanto uma rede convolucional com *kernel* de $7x7$ utilizaria 147 pesos para cada neurônio da primeira camada.

A simplificação dos dados também é responsável por melhorar a eficiência da rede neural em extrair características espaciais da entrada. Já que um *kernel* é utilizado para processar os dados, apenas uma região é analisada por vez, excluindo conexões desnecessárias entre *features* espacialmente distantes, e realçando relações entre *features* vizinhas. Isso não só acelera o treinamento, mas também aumenta o desempenho durante testes. É importante ressaltar que a relação espacial entre dados não necessariamente significa distância, a posição em que dados são fornecidos como entrada também pode representar relações temporais, ordinárias ou até correlações entre *features* [53].

2.3.4 Redes Temporais Convolucionais

Redes Temporais Convolucionais (TCN, do inglês *Temporal Convolutional Network*) são uma evolução das redes convolucionais para serem melhor aplicadas como redes recorrentes. Atualmente, redes recorrentes como LSTM e GRU são amplamente utilizadas

para problemas de séries temporais em razão da possibilidade de armazenar um estado entre operações. Esse estado não guarda toda a informação recebida anteriormente, mas somente a parte relevante para a rede neural naquele momento. Por isso, boa parte da informação de longo prazo é descartada conforme os dados armazenados são atualizados, tornando mais difícil para a rede neural encontrar relações entre dados distantes entre si. As redes temporais convolucionais procuram resolver esse problema utilizando dilatações [54].

As dilatações são uma forma diferente de aplicar uma convolução. O valor de dilatação é o espaço percorrido entre multiplicações durante o processo de uma convolução. Assim, uma convolução tradicional possui valor de dilatação 1, enquanto uma TCN utiliza valores exponenciais para encadear convoluções. A Figura 2 (Adaptada de [55]) apresenta exemplos de dilatação [56]. Nessa figura, as dilatações são iniciadas em 1 e seguidas por 2 e 4, o que possibilitaria que informações de 8 entradas (laranja) fossem representadas em apenas 1 (vermelha).

Utilizando as dilatações, é possível entregar uma entrada maior para a rede neural sem aumentar significativamente a complexidade do processamento da rede, possibilitando que, para cada iteração da rede neural, informações completas do passado sejam recebidas [57]. É importante ressaltar que nesse tipo de rede cada elemento da entrada influencia na saída, como apresentado na Figura 2. Isso evita o problema de descarte de informação sem que seja necessário mais poder de processamento, nem uma unidade de memória para armazenamento de estado.

2.4 Lógica Difusa (*Fuzzy*)

A lógica difusa foi proposta em 1965 e, desde então, é utilizada como uma solução para problemas que requerem tolerância à incerteza. Ela oferece uma alternativa à lógica binária de verdadeiro e falso, ao permitir a utilização de um gradiente de possibilidades. Para tanto, funções de pertinência são utilizadas para definir a categoria de cada dado, de forma que cada função retorne um valor entre 0 e 1, em que 0 significa completamente não relacionado e 1, totalmente relacionado [58]. Essa classificação utiliza termos linguísticos para melhor compreensão, além de servir como saída para sistemas que precisam comunicar tal incerteza, como robôs que precisam demonstrar a razão e conhecimento de suas decisões [59].

Uma outra aplicação da lógica difusa é a unificação de entradas para uma tomada de decisão. Quando diversos fatores podem influenciar um resultado, a lógica difusa pode ser utilizada para atribuir diferentes pesos para cada elemento. Um exemplo simples é o sistema de avaliação de restaurantes em que os elementos que influenciam na nota são: a qualidade de atendimento, o sabor da comida, e o preço. Utilizando as funções

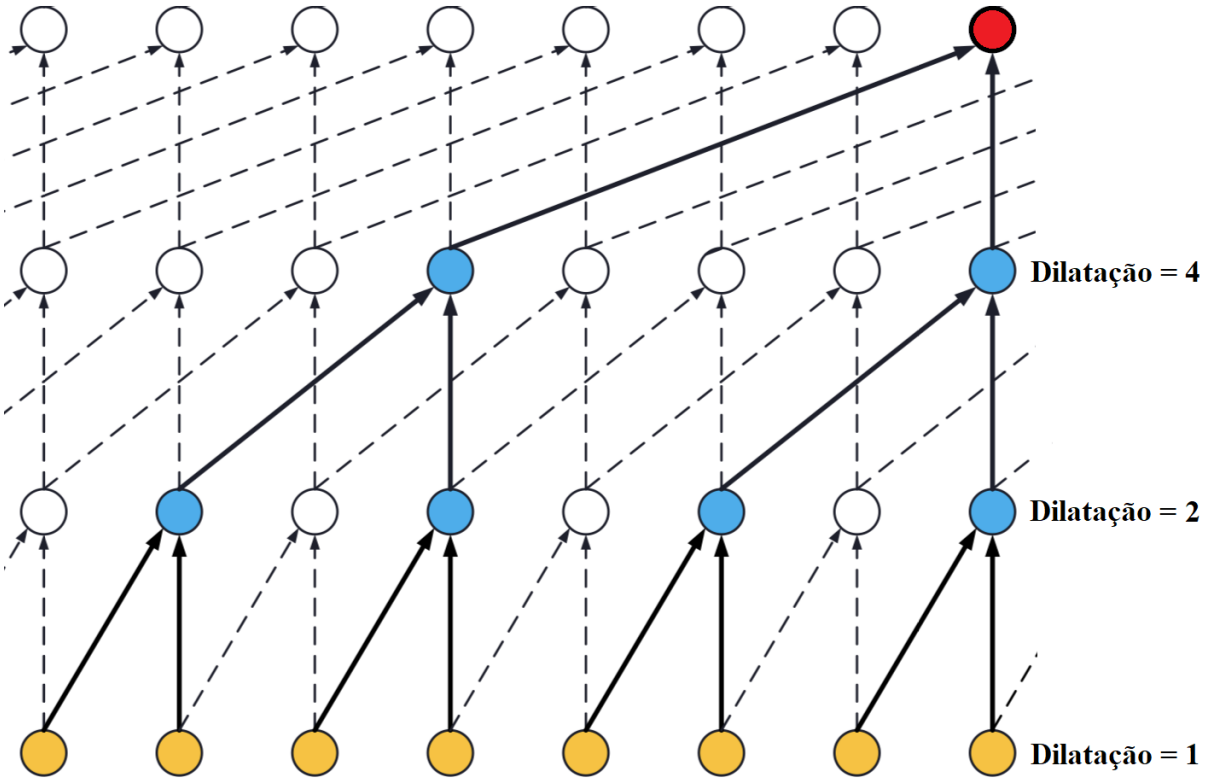

Figura 2 – Dilatações de valores 1 e 2 e 4. Fonte: Adaptado de Rémy 2019

de pertinência corretas, é possível valorizar mais o sabor ao mesmo tempo que penaliza notas muito ruins de atendimento sem dar valores excessivos de pontos para atendimentos ótimos. O sistema também pode contar com uma tolerância maior para preços mais baixos enquanto é mais exigente para preços mais altos.

Para exemplificar, uma função de pertinência será apresentada para cada parâmetro. Começando pelo preço, uma função logística pode ser utilizada para entregar pontos enquanto os preços forem muito baixos, de forma que compensaria por uma qualidade de atendimento ou um sabor menor. A função representada em 2.8 e na Figura 3, é uma função similar à 2.6 já apresentada, porém decrescente, deslocada para a direita e com valor máximo de 1.5. Dessa forma, um restaurante com preços baixos receberá até 1.5 pontos que diminuem conforme o preço aumenta.

$$f(x) = \frac{1.5}{1 + e^{4(x-1.2)}} \tag{2.8}$$

A segunda função representa a pontuação de qualidade de atendimento. Representada pela equação 2.9 e Figura 4 está uma função que utiliza uma função de reta $y = x$ e uma função também similar à 2.6. Essa função penaliza um atendimento ruim com pontos negativos. Um atendimento mediano consegue gradativamente mais pontos, mas um atendimento excelente não recebe tantos pontos a mais já que não deve ser a única

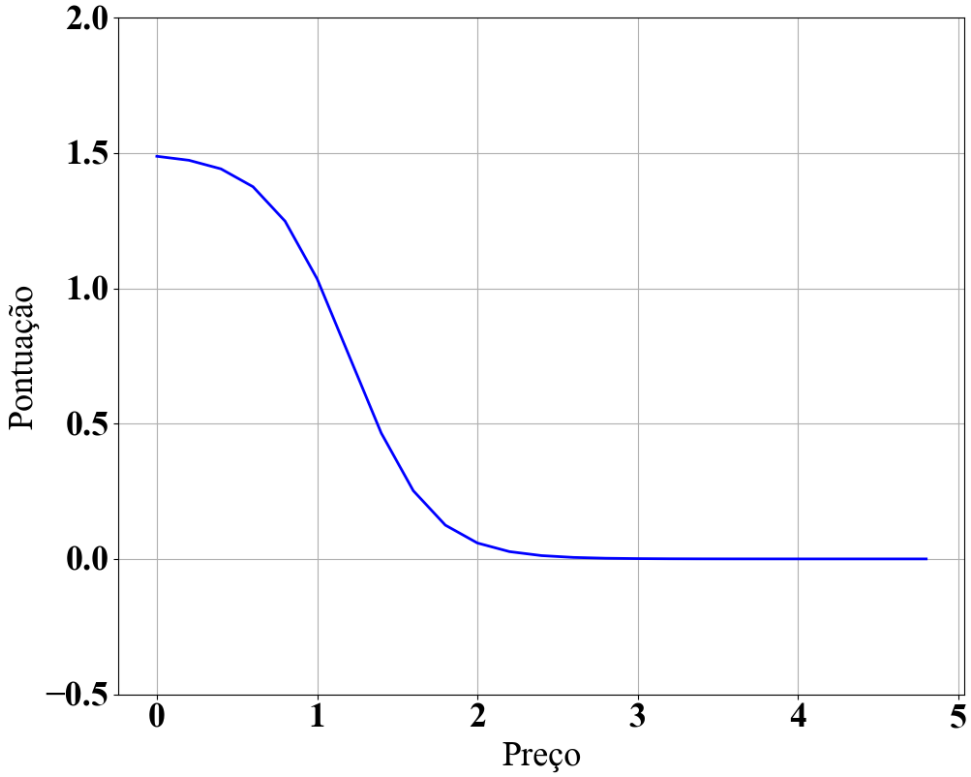

Figura 3 – Função de pontuação por preço

razão de uma boa avaliação em um restaurante.

$$g(x) = max(min(x - 1; \frac{1}{1 + e^{-4(x-2.5)}}); -1) \tag{2.9}$$

Por fim, a função que dá pontos pelo sabor está representada na equação 2.10 e na figura 5. Essa função é metade de uma função sino subtraída de cinco. Essa função faz com que o sabor seja o que mais daria pontos para um restaurante, mas sem tirar a necessidade de um bom atendimento, já que com o melhor sabor apenas 4 pontos seriam concedidos.

$$h(x) = 5 - \frac{5}{1 + |\frac{x^{2.5}}{2}|} \tag{2.10}$$

Dessa forma, uma aplicação poderia utilizar as três funções de pertinência apresentadas para criar uma avaliação de restaurantes baseada em pontos. É comum que aplicações de lógica difusa também apliquem o processo de "*defuzzyficação*", que significa transformar o valor resultante de volta para um resultado discreto. Esse processo pode utilizar um ou mais limiares para separar os valores de cada resultado. Seguindo o exemplo da avaliação de restaurante, os limiares poderiam separar igualmente os valores de 0 a 5 entre cinco classificações: péssimo, ruim, mediano, bom, e ótimo.

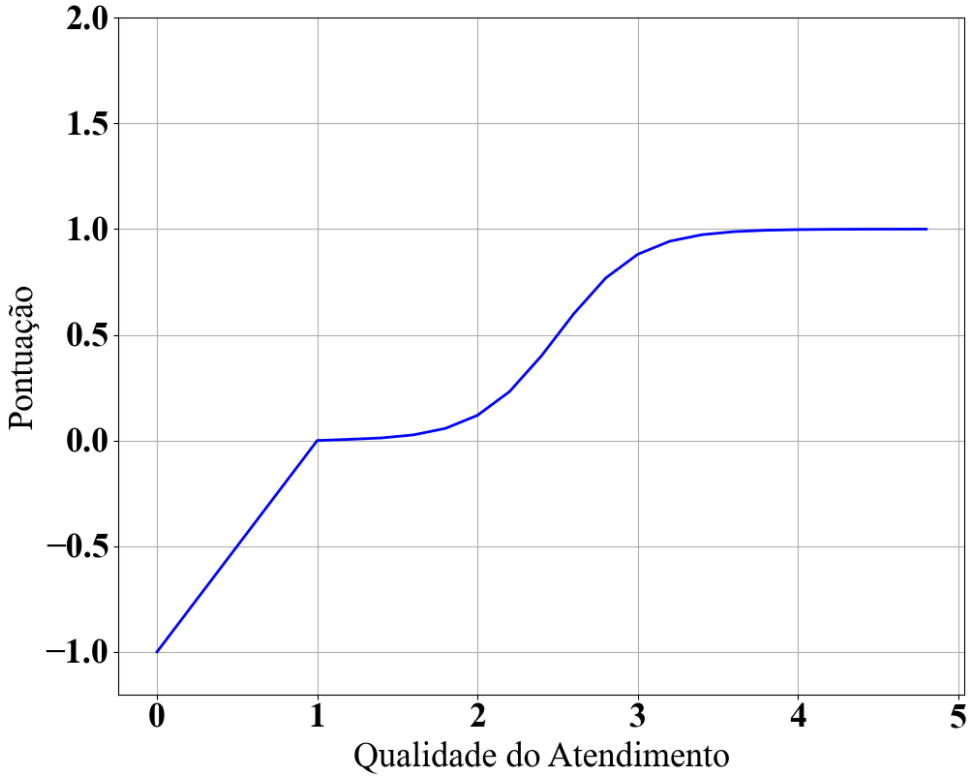

Figura 4 – Função de pontuação por qualidade do atendimento

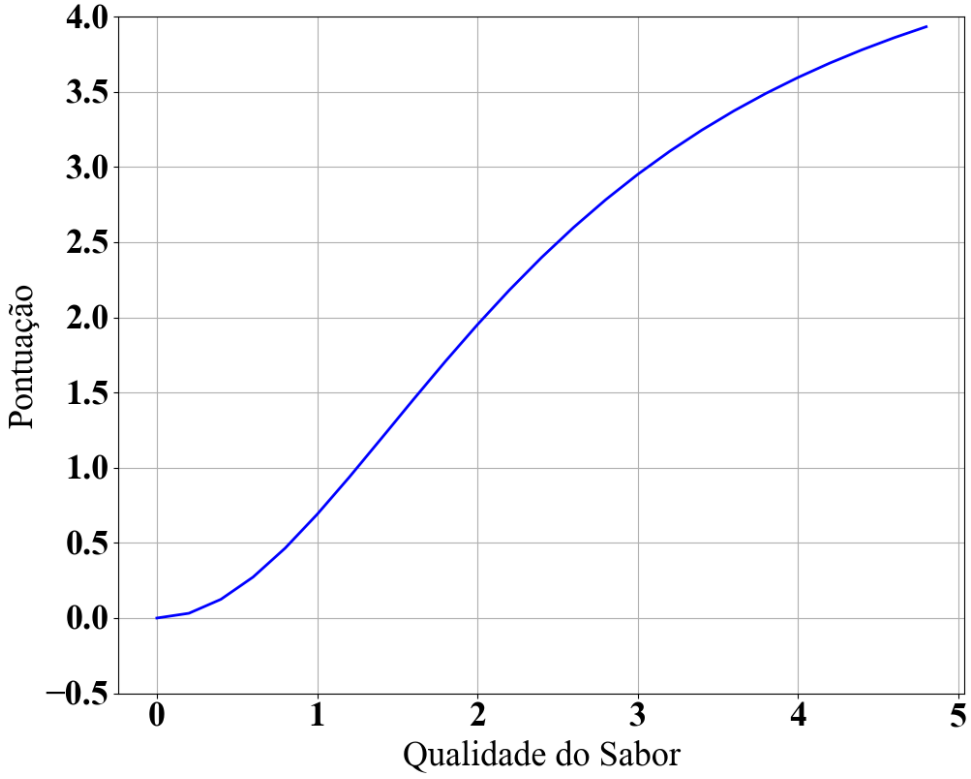

Figura 5 – Função de pontuação por qualidade do sabor

3 TRABALHOS RELACIONADOS

Diferentes formas de aprendizagem de máquina já foram estudadas para detectar e mitigar ataques cibernéticos. Dentre essas, tem se tornado cada vez mais comum a utilização de aprendizagem profunda, seja para detectar anomalias ou para extrair *features* dos dados de tráfego. Além disso, a abordagem ao problema também varia consideravelmente com o sistema implementado, seja por basear-se em assinatura ou anomalia, seja por utilizar aprendizagem supervisionada ou não, seja por analisar pacotes ou fluxos. Nesse capítulo serão destacadas algumas dessas abordagens.

No trabalho de Mihoub *et al.* [60], um sistema com dois módulos principais é apresentado. O primeiro módulo é o detecção, que fornece dados precisos sobre o tipo de ataque e o tipo de pacote atacante ao módulo de mitigação para que as medidas mais apropriadas sejam tomadas. Para fazer isso, um algoritmo de *Random Forest* é aplicado e comparado com outros tipos de aprendizagem de máquina. O artigo tem um foco em proteção de internet das coisas, por isso o conjunto de dados IoT-Bot foi utilizado para validar o modelo proposto. O dataset possui apenas ataques de DoS e DDoS, por isso outros tipos de ataques não foram testados

Zeng *et al.* [61] apresentam um sistema que detecta ataques DDoS utilizando "raciocínio causal" para identificar anomalias. Para validar o modelo, o conjunto de dados CICDDoS2019 foi utilizado e o algoritmo CICFlowMeter foi aplicado para extrair suas *features*. Diferentes tipos de ataque DDoS foram aplicados para testar o modelo e todos apresentaram resultados satisfatórios.

Para possibilitar que pequenas empresas se defendam contra ataques de DDoS, Gaurav *et al.* [62] propõem um sistema simples de detecção de ataques utilizando *random forest*, máquina de vetor de suporte, regressão logística e outros métodos de aprendizagem de máquina para detectar ataques. O simulador de eventos OMNET++ foi utilizado para gerar um conjunto de dados de tráfego em uma rede. Apesar de não superar o estado da arte, os autores alegam que sistema apresenta desempenho satisfatório em relação ao seu custo. Entretanto, o conjunto de dados não é amplamente utilizado pela comunidade científica, dificultando sua comparação com outros modelos.

Manasrah *et al.* [63] criaram um sistema de detecção de *botnets* que evita ataques de geração de domínio. O algoritmo analisa domínios criados para esconder *botnets* e encontra anomalias em seus nomes. Para fazer a detecção, os métodos *K-nearest neighbor*, máquina de vetor de suporte, árvore de decisão, e rede neural artificial foram aplicados. A rede neural superou os outros métodos. O modelo foi avaliado utilizando um conjunto de dados gerado pelos próprios autores. Apesar de não necessariamente prevenir ataques,

os autores apresentam uma forma que pode bloquear o funcionamento de *botnets*.

Assis *et al.* [64] utilizaram análise de fluxos para detectar e mitigar diversos ataques de rede. Para isso, uma rede neural GRU foi treinada para separar fluxos maliciosos de normais. O sistema foi comparado com redes neurais convolucionais, redes neurais LSTM, máquina de vetor de suporte e outros. Para validar o sistema, dois conjuntos de dados públicos foram utilizados: CICIDS2018 e CICDDoS2019. Além de um resultado de classificação satisfatório, o modelo apresentou alto desempenho de fluxos por segundo. Apesar de resultados promissores, o método requer que o ataque seja apresentado ao modelo durante o treinamento, o que pode dificultar a detecção e mitigação de ataques novos.

Uma revisão de trabalhos foi apresentada no artigo de Martins *et al.* [65]. São abordados nesse estudo conceitos, desafios e taxonomia de sistemas de detecção de anomalias. Além disso, alguns sistemas foram testados para verificar sua eficiência em tempo real, utilizando dados da lista de *Common Vulnerabilites and Exposures* (CVE). O estudo conclui que os sistemas de detecção de intrusão mais comuns são vulneráveis a ataques mais novos, por terem seu foco voltado a serviços e aplicações mais frequentemente utilizados.

Chatterjee *et al.* [66] fizeram um *survey* que reúne trabalhos de detecção de anomalias em rede desde 2019. Com o foco em internet das coisas, esse estudo investiga os avanços, os métodos aplicados, as dificuldades e futuros trabalhos sobre detecção de anomalias. Uma das conclusões do trabalho é de que a área de detecção de anomalias específica para IoT ainda está começando a avançar para soluções mais complexas e próprias para esse tipo de rede.

No trabalho de Sgueglia *et al.* [67], uma revisão de artigos sobre detecção de anomalias é feita. Os focos dessa revisão são verificar principalmente as formas de redução de dimensionalidade aplicadas, formas de monitoramento em tempo real, os datasets aplicados para validação e os cenários reais utilizados para teste. O trabalho conclui que existe uma falta de trabalhos que alcancem a detecção em tempo real, que há uma carência de técnicas avançadas de redução de dimensionalidade, e que os conjuntos de dados disponíveis normalmente são compostos por dados sintéticos e podem causar uma falsa percepção de eficiência dos modelos testados.

Qingfeng *et al.* [68] aprestaram um sistema de detecção de anomalias utilizando redes neurais LSTM combinadas com redes de convolução de grafo. Nesse sistema, a rede de convolução extrai relações espaciais dos dados, enquanto a LSTM extrai as temporais. O sistema foi testado em dois conjuntos de dados reais: CICIDS2017 e CTU-13. O primeiro é composto por pacotes capturados de uma rede tradicional e o segundo é um tráfego de *botnet*. O sistema proposto apresenta resultados satisfatórios, mas foi concluído que um tráfego maior aumenta o tamanho do grafo e, consequentemente, o tempo de detecção.

Um sistema de detecção de anomalias utilizando clusterização foi proposto por Chonghua *et al.* [69]. O foco desse estudo é detectar anomalias coletivas (*Collective Anomaly*) que, em uma breve explicação, são um conjunto de eventos que, individualmente, não são anômalos, mas sim quando ocorrem ao mesmo tempo. Assim, o sistema desenvolvido analisa múltiplos *clusters* para detectar a anomalia, diferente do que outros sistemas fazem. Para validar o modelo, os conjuntos de dados CICIDS2017 e UNSW-NB15 foram utilizados.

Derhab *et al.* [70] utilizaram redes neurais temporais convolucionais para criar um *framework* de detecção de anomalias. Os autores também utilizam uma técnica de *oversampling*, chamada *smote-nc*, para diminuir os efeitos de um *dataset* desbalanceado como o Bot-IoT, utilizado para validar o modelo. Um aspecto ressaltado pelos autores é a redução das *features* utilizadas, que diminui o consumo de memória em 77%, além de acelerar o treinamento da rede neural.

Por fim, Novaes *et al.* [71] utilizam redes neurais generativas adversárias (GAN) para detectar anomalias em uma rede. Um dos focos do estudo é proteger o modelo contra ataques adversariais, que são perturbações propositais no tráfego, para causar erros de classificação no sistema. O conjunto de dados CICDDoS2019 foi utilizado para avaliar o modelo.

Além dos trabalhos de detecção de anomalias em redes já mencionados, também vale citar artigos em que aprendizagem profunda foi utilizada para detecções em outras áreas, como o trabalho de Audibert *et al.* [72]. Nesse trabalho, é feita uma análise de redes neurais profundas em comparação com aprendizagem de máquina tradicional para detectar anomalias em séries temporais. Para isso, são comparadas dezesseis técnicas diferentes em cinco conjuntos de dados. Uma das conclusões do estudo é que os métodos tradicionais tendem a superar os profundos quando poucos dados estão disponíveis. Por outro lado, os modelos de aprendizagem profunda se apresentaram superiores ao detectar anomalias contextuais em um dos conjuntos de dados, o que não representa evidência concreta, mas pode inspirar alguns estudos no tema.

Outro trabalho relevante, e que não necessariamente foi aplicado em redes de computadores, é a pesquisa feita por Lindemann *et al.* [73]. Esse trabalho tem foco em redes neurais *long short-term memory* (LSTM). Esse tipo de rede é frequentemente aplicado em problemas de séries temporais, por possuir um módulo de memória para armazenar contexto. No estudo, foram compilados trabalhos de detecção de anomalias em redes de computadores, em imagens, em maquinário industrial, trajetória de pedestres, e outros. Esse trabalho aborda formas de arquitetura de rede, abordagens de detecção e tendências em trabalhos recentes.

Por fim, um resumo dos trabalhos de detecção de anomalias em rede é apresentado na tabela 1

Tabela 1 – Resumo trabalhos relacionados. Fonte: o próprio autor

Autor	**Dados Utilizados**	**Ambiente**	**Principais Características**
Mihoub *et al.* [60]	IoT-Bot	Redes IoT	Algoritmo de *Random Forest* utilizando abordagem *look-back* que identifica o tipo de ataque e o mitiga.
Zeng *et al.* [61]	CICDDoS2019	Redes Tradicionais	Algoritmo de "raciocínio causal" para detecção e CICFlowMeter para extração de *features*.
Gaurav *et al.* [62]	Gerados com OMNET++	Redes Tradicionais Pequenas	Algoritmo de baixo custo computacional para detectar ataques DDoS.
Manasrah *et al.* [63]	Dados gerados pelos autores	Redes de DNS	Algoritmo de detecção de anomalias em nomes de DNS para prevenir *botnets* de se conectarem ao controle central.
Assis *et al.* [64]	CICIDS2018 e CICDDoS2019	Redes SDN	Algoritmo de aprendizagem profunda GRU para detectar e mitigar cyberataques validado por conjuntos de dados públicos.
Martins *et al.* [65]	Lista CVE	Redes IoT	Revisão de sistemas de detecção de intrusão, com testes em tempo real de sistemas de *software* aberto mais utilizados.
Chatterjee *et al.* [66]	Não se aplica	Redes IoT	Um *survey* sobre trabalhos de detecção de anomalias específico para Internet das Coisas.
Sgueglia *et al.* [67]	Não se aplica	Redes IoT	Uma revisão de trabalhos de detecção de anomalias em redes, realçando limitações e problemas a serem resolvidos em trabalhos futuros.
Qingfeng *et al.* [68]	CICIDS2017 e CTU-13	Redes Tradicionais e *Botnet*	Utiliza LSTM com convolução de grafo para detectar anomalias em rede.
Chonghua *et al.* [69]	CICIDS2017 e UNSW-NB15	Redes tradicionais	Propôs um algoritmo de clusterização para detectar anomalias principalmente coletivas.
Derhab *et al.* [70]	Bot-IoT	Redes IoT	Sistema de detecção de anomalias que utiliza redes neurais temporais convolucionais.
Novaes *et al.* [71]	Dados emulados na UEL e CICDDoS2019	Redes Definidas por *Software*	Sistema de detecção de intrusão, utilizando redes neurais generativas adversárias

4 SISTEMA PROPOSTO

4.1 Coleta de fluxos

Como já mencionado, o sistema proposto utiliza redes definidas por *software* para coletar dados de tráfego e alimentar um modelo de aprendizagem profunda que detecta anomalias a cada segundo. Esses dados são coletados pelo controlador, possivelmente utilizando o protocolo *Openflow*, em forma de fluxos. Fluxos são um resumo de uma interação entre dois endereços de *IP* e porta. Neles estão contidas diversas informações, como endereço de origem e destino, quantidade de pacotes de ida e volta, tamanho médio dos pacotes, e duração do fluxo. A tabela 2 apresenta alguns exemplos de fluxos.

Além do protocolo *Openflow*, existem outras formas de se coletar dados de tráfego. O segundo cenário desse estudo utiliza dados gerados e analisados pelo CICFlowMeter, uma aplicação desenvolvida pela Universidade de Nova Brunswick. Nesse caso, os dados de cada fluxo contêm mais de 80 características que poderiam ser utilizadas, dentre elas: protocolo de comunicação, número de pacotes em cada direção, tamanho mínimo e máximo de pacotes em cada direção, quantidade de *flags* de cada tipo, e outras, incluindo as presentes no *Openflow*.

Para alimentar o sistema de detecção de anomalias, os fluxos são agrupados pelo segundo em que iniciaram e então encaminhados para o pré-processamento. Nessa etapa, as seis dimensões de fluxos utilizadas pelo sistema são extraídas. São elas: bits por segundo, pacotes por segundo, entropia de *IP* de origem, entropia de *IP* de destino, entropia de porta de origem, e entropia de porta de destino.

Tabela 2 – Exemplos de fluxos

Início do Fluxo	IP Origem	Porta Origem	IP Destino	Porta Destino	Quantidade de Pacotes	Bytes	Duração em ns
2000-01-05 09:15:44	27.87.82.44	80	10.0.0.55	5078	4	868	90000000
1979-10-31 15:35:19	148.140.24.70	80	10.0.0.45	4000	2	384	688000000
1965-04-07 00:00:00	22.87.124.99	53	10.0.0.31	4000	2	584	687000000

4.2 Previsão e Detecção de Anomalias

A etapa de detecção do sistema utiliza seis redes neurais, que são treinadas para que cada uma preveja uma das seguintes dimensões de fluxo: bits por segundo, pacotes

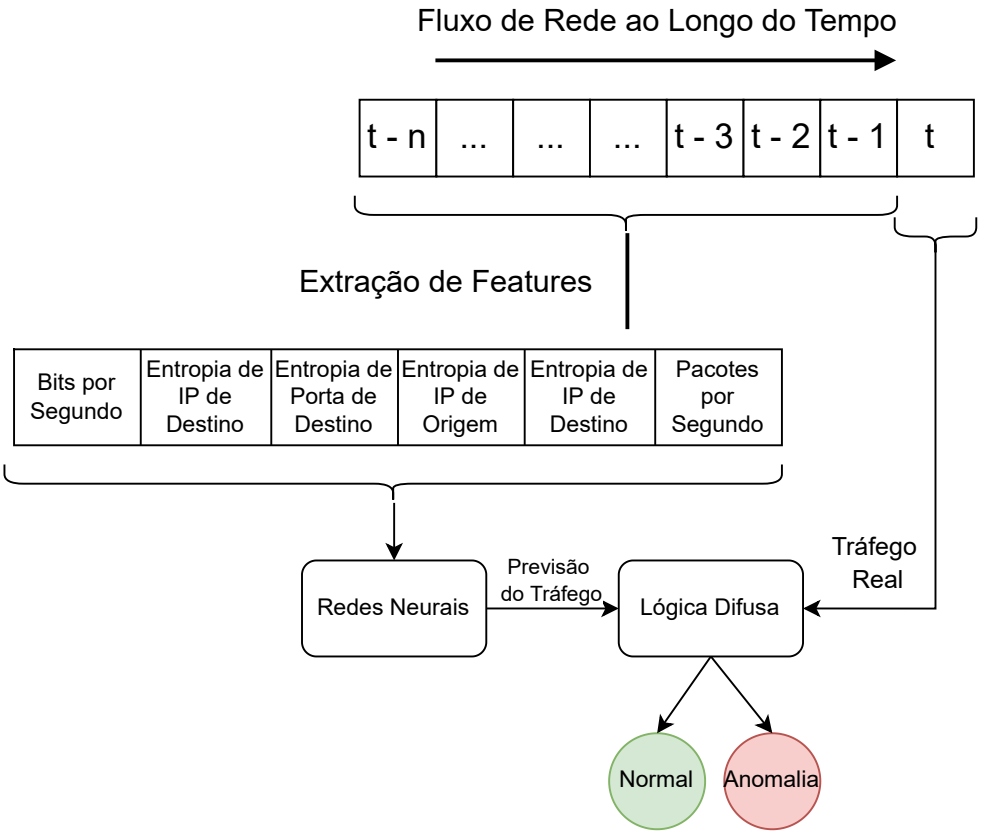

Figura 6 – Modelo Proposto

por segundo, entropia de *IP* de origem, entropia de *IP* de destino, entropia de porta de origem, e entropia de porta de destino. Cada previsão é comparada com o tráfego real da rede em uma função de pertinência de lógica *fuzzy*. Por fim, um limiar de "*defuzzyficação*" é utilizado para classificar o momento como anômalo ou normal. Um resumo desse modelo é apresentado na Figura 6.

A entropia utilizada é a de Shannon, representada na equação 4.1. Nessa equação, H representa a entropia, i representa cada elemento individual, pi indica a probabilidade desse evento acontecer, e N é a quantidade de eventos possíveis. A entropia é um conceito utilizado em diversas áreas da física e representa o grau de desordem ou aleatoriedade de um sistema [74]. Quanto maior a entropia, maior a incerteza e aleatoriedade do sistema. Por exemplo, um dado de seis lados tem uma entropia menor do que um dado de vinte, já que a incerteza deste é maior.

$$H = -\sum_{i=1}^{N} p_i \log_2(p_i) \tag{4.1}$$

Ao calcular a entropia para as dimensões de fluxo, o valor apurado representa a incerteza presente nas ocorrências de um segundo de tráfego. Por exemplo, a entropia de *IP* de origem representará o grau de incerteza das ocorrências de endereços *IP* em cada

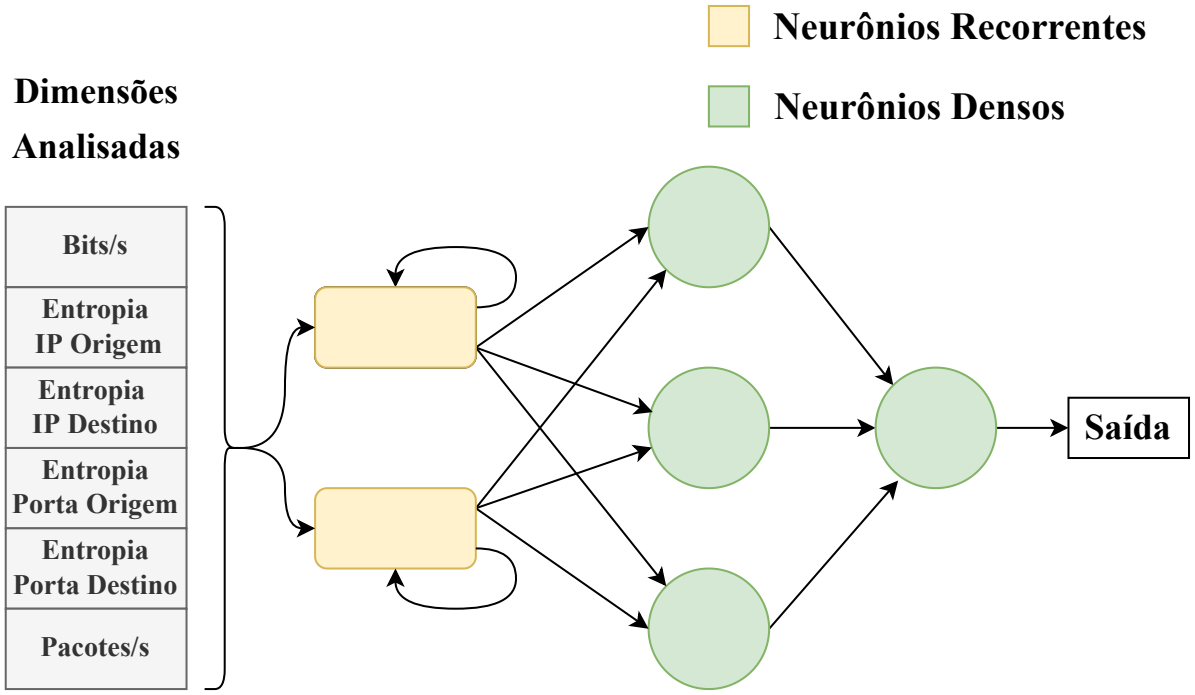

Figura 7 – Padrão de Arquitetura Utilizado na Rede Neural

fluxo. Se uma grande quantidade de endereços começar a enviar pacotes para a rede, a entropia aumenta. Da mesma forma, se um endereço começar a enviar consideravelmente mais pacotes que os outros, a entropia diminui.

O cálculo de entropia tem duas funções principais. A primeira é representar o tráfego da rede durante um período de tempo para a rede neural, sem que seja necessário alimentá-la com cada um dos fluxos presentes no intervalo. A segunda consiste em converter informações que não têm nenhuma relação numérica, como endereços de *IP* e porta, para um valor numérico útil para uma rede neural. Não tem relevância se endereço de *IP* possui um valor maior que outro, pois a relação entre dois endereços não é quantitativa, mas sim qualitativa. Por outro lado, a entropia dá um valor quantitativo à dimensão, tornando-a útil para a rede neural.

Para esse trabalho, o tipo de rede neural utilizado foi o *Gated Recurrent Unit* por ser desenvolvido para séries temporais. A arquitetura da rede utilizada segue um padrão, que tem neurônios GRU na camada de entrada e neurônios tradicionais (também chamados de densos) nas subsequentes, como mostrado na Figura 7. O *input* de cada uma das redes é composto pelos últimos cinco segundos de todas as dimensões de fluxo. Outros hiperparâmetros (ajustes que afetam o desempenho do modelo) são detalhados na seção 4.4. Os dados são normalizados com o uso do *Standard Scaler* da biblioteca Scikit Learn [75]. É comum a utilização de normalização para melhorar performance de um modelo, ao mesmo tempo que ajusta o alcance de *output* para funções de ativação.

Para o treinamento da rede neural, a biblioteca Keras para Python foi utilizada com o otimizador Adam com parâmetros padrões [76]. Outros hiperparâmetros do sistema serão discutidos no capítulo de estudo de caso. Durante o treinamento, apenas dados de tráfego normal foram utilizados. Assim, é esperado que, ao ser apresentada a dados anômalos, a rede resulte em um maior erro de previsão e, consequentemente, seja detectada uma anomalia. Depois de treinar a rede neural, um dos dias com ataque é testado e um limiar de decisão é escolhido de forma que separe os dados normais dos anômalos. Esse limiar de decisão é utilizado na "*defuzzyficação*" dos dados e também é chamado de limiar *fuzzy*.

Para fazer a detecção, a função de pertinência de lógica *fuzzy* utilizada é a gaussiana, representada na função 4.2 e na Figura 8. Nesse processo, o erro da previsão é inserido na função, a fim de que seja avaliado o quão provável é que aquele tráfego seja normal, baseado no desvio padrão dos últimos segundos. Um desvio maior resulta numa função mais tolerante. Na função 4.2, x representa o tráfego real, y a previsão da rede neural, σ o desvio padrão dos últimos 20 segundos. A constante 4,47 foi escolhida baseada no uso da inequação de Chebyshev [77], descrita na equação 4.3, em que X_d representa o dado, μ a média dos dados, σ o desvio padrão dos dados e k é a distância em desvios padrões da média. Assim, dado um k, a inequação retorna o grau de certeza de que o dado X_d pertence ao conjunto de dados. Portanto, é a probabilidade de um dado pertencer ao conjunto analisado. Nesse trabalho, o grau de confiança alvo é de 95%, que implica em um $k \simeq 4,47$. Nesse cenário, função gaussiana foi escolhida por ser contínua e por possuir uma inclinação leve em seu centro, que aumenta conforme se afasta dele. Isso significa que erros pequenos e esperados tendem a ser ignorados, enquanto erros maiores são acentuados pela função.

$$f(x) = 1 - e^{\frac{-(x-y)^2}{2(4,47\sigma)^2}} \tag{4.2}$$

$$P(|X_d - \mu| \leq k\sigma) \geq \left(1 - \frac{1}{k^2}\right) \tag{4.3}$$

O Algoritmo 1 descreve o processo de detecção, onde X_t e X_{t-1} representam as dimensões de fluxo do segundo analisado e dos segundos anteriores, respectivamente. Os y_n representam a previsão de cada uma das redes neurais e são enviados para a função de pertinência, que retorna um número. O número é então comparado com o limiar encontrado durante o treinamento. Caso seja maior o valor será considerado anômalo, e normal caso o contrário.

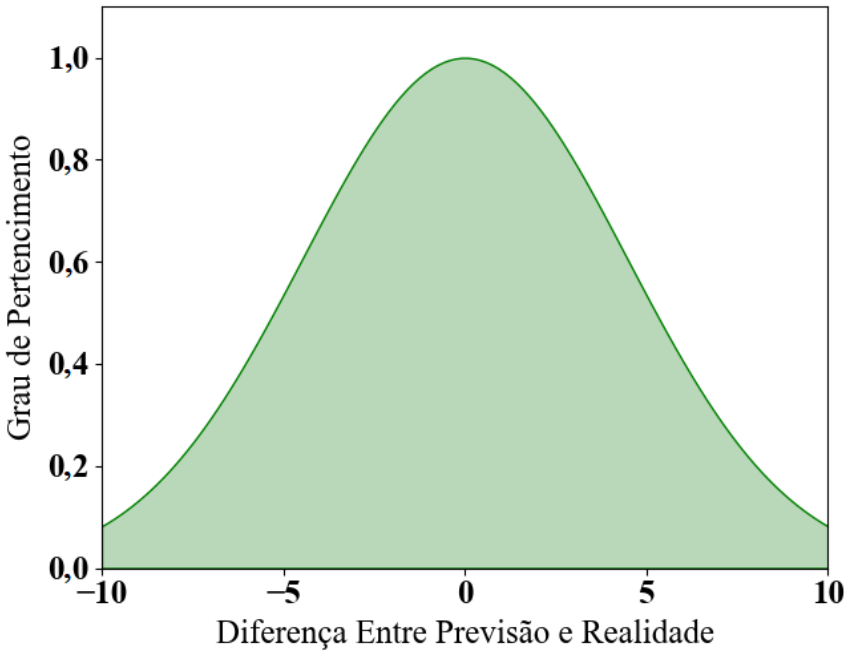

Figura 8 – Função de Pertinência quando desvio padrão é igual a 1

Algoritmo 1 Algoritmo de Detecção de Anomalias

Require: $X_{t-1} \leftarrow$ (x1, x2, x3, x4, x5, x6); limiar_fuzzy
Ensure: É normal ou Anômalo

```
y1 ← Gru_Bits(x1, x2, x3, x4, x5, x6)
y2 ← Gru_Packets(x1, x2, x3, x4, x5, x6)
y3 ← Gru_IP_Src_entropy(x1, x2, x3, x4, x5, x6)
y4 ← Gru_Port_Src_entropy(x1, x2, x3, x4, x5, x6)
y5 ← Gru_IP_Dst_entropy(x1, x2, x3, x4, x5, x6)
y6 ← Gru_Port_Dst_entropy(x1, x2, x3, x4, x5, x6)
Y ← (y1, y2, y3, y4, y5, y6)
fuzzy ← função_pertinencia_fuzzy(Y, X_t)
if fuzzy > limiar_fuzzy then
    return Anômalo
else
    return Normal
```

4.3 Mitigação

O processo de mitigação faz uso da alta capacidade de configuração que o plano de controle das redes definidas por *software* oferece. Pelo plano de controle, é possível implantar políticas de encaminhamento que bloqueiam determinados endereços *IP*. Dessa forma, esse sistema pode mitigar ataques, independentemente de ação humana, ao detectá-los. Para esse módulo do sistema funcionar, são necessários o resultado de detecção do módulo anterior, o tipo de ataque detectado e o conjunto de fluxos que dispararam o alarme. Para identificar o tipo de ataque, o sistema detecção analisa quais entropias são mais afetadas pelo ataque. Ataques *portscan* farão a entropia de porta de destino subir, enquanto ataques distribuídos de negação de serviço (DDoS) diminuirão a entropia de porta de destino e aumentarão a entropia de porta de origem.

O Algoritmo 2 demonstra como o processo de mitigação é feito. O algoritmo utiliza uma "*Safe List*", que evita que tráfego legítimo seja bloqueado devido a falsos positivos do módulo de detecção. Assim, quando não há ataques, a lista recebe os endereços de origem do tráfego e os guarda por 5 minutos. Durante esse tempo, se um ataque é detectado, os endereços na lista não são bloqueados. Quando um *IP* é considerado malicioso e não está na lista segura, ele é bloqueado por 20 segundos.

Algoritmo 2 Algoritmo de Mitigação

```
1: if Ataque Portscan then
2:     IP atacado ← Endereço de IP que recebe mais fluxos
3:     IP Malicioso ← Dos fluxos enviados para o IP atacado, escolher o IP que possui
   maior variedade de portas
4:     if IP não está na Safe List then
5:         Bloquear fluxos do IP malicioso
6: else if Ataque DDoS then
7:     IP atacado ← Endereço de IP que recebe mais pacotes
8:     Porta atacada ← Do IP atacado, pegar a porta que recebe mais fluxos
9:     Lista de IPs suspeitos ← Todos os IPs que têm como destino para o IP atacado e
   Porta atacada
10:    for IP na Lista de IPs suspeitos do
11:        if IP não está na Safe List then
12:            Bloquear pacotes do IP
```

4.4 Hiperparâmetros do Sistema

Durante o desenvolvimento do sistema, algumas de suas possíveis configurações foram testadas para maximizar sua performance. Essas configurações são frequentemente chamadas de hiperparâmetros. Nessa seção, uma explicação sobre cada um dos hiperparâmetros estudados e como cada um pode afetar o desempenho do modelo proposto:

1. Quantidade de segundos de entrada: esse valor é importante por definir quanto contexto será entregue à rede neural para que uma previsão do próximo segundo seja realizada. Um número baixo significaria informações insuficientes para uma previsão, enquanto um número alto atuaria como um ruído para a rede neural, o que diminuiria sua precisão;

2. Quantidade de neurônios nas camadas GRU e densas: mais neurônios significam mais formas de se processar a informação recebida e extrair conhecimento relevante dos dados. Neurônios demais causarão *overfitting*, já que serão necessários dados demais para treiná-los sem que a capacidade de generalização da rede neural seja perdida;

3. Quantidade de camadas densas da rede: a variação desse parâmetro afeta a complexidade das informações extraídas dos dados. Maior número de camadas permite que os dados inseridos passem por mais transformações e, consequentemente, maior a possibilidade de extrair padrões das informações. No entanto, camadas em excesso causam um processo de treinamento mais lento e requerem mais dados para alcançar o mesmo resultado obtido com menos camadas.

4. Tamanho do "*Batch*" de treinamento: é a quantidade de entradas do conjunto de treino utilizadas antes de cada atualização dos pesos da rede neural. Um valor pequeno resulta em uma má estimativa do gradiente, já que o subconjunto pode não representar o conjunto total de forma genérica. Um valor grande demais resultará na necessidade de um treinamento mais longo, já que os pesos da rede serão atualizados menos vezes por época;

5. A janela *fuzzy*: durante a detecção, o sistema compara a previsão com o tráfego real. Para isso, o desvio padrão dos últimos segundos é utilizado como uma forma de aumentar ou diminuir a tolerância a erros da comparação de forma que variações maiores aumentam a tolerância. Valores muito altos podem aumentar a tolerância exageradamente, enquanto valores baixos podem resultar em falsos positivos durante a ocorrência de "ruídos" no tráfego.

Para a escolha dos hiperparâmetros, buscou-se maximizar a métrica *F1-score* da equação 4.6. Essa métrica é uma média harmônica das métricas "precisão" (representada na equação 4.4) e "revocação" (representada na equação 4.5). Esse tipo de métrica é útil quando existe um desbalanceamento nos dados, o que tende a ser o caso para esse tipo de problema, já que ataques à rede não são tão frequentes.

A métrica de precisão indica a proporção de acertos dentre todos os elementos que foram apontados como uma determinada classe. Dessa forma, essa métrica diminui quando o modelo aponta elementos de outras classes como sendo da primeira. Apesar de

útil, essa métrica não deve ser utilizada sozinha, principalmente em casos em que há um desbalanceamento entre classes. Em um exemplo, pode ser que apenas uma parte exclusiva do conjunto total da classe seja apontada como pertencente a ela sem qualquer elemento de outra. Isso resultaria em uma precisão perfeita, mas o modelo ainda poderia melhorar. Portanto, essa métrica é uma boa forma de determinar o grau de confiança em um modelo quando aponta para determinada classe, já que, mesmo em casos como o exemplo citado, uma precisão alta significa que, quando um elemento é apontado como pertencente a uma classe, há uma chance grande de que o modelo esteja certo, independentemente de outros erros.

A revocação indica a proporção de elementos que foram apontados como determinada classe dentre todos os elementos pertencentes a ela. Essa métrica age como um contraponto à precisão, já que, no exemplo dado anteriormente, seria muito baixa. Contudo, da mesma forma que é possível abusar da precisão, também é possível se aproveitar da revocação. Um exemplo disso seria um modelo que apontasse todos os elementos de um conjunto de dados como pertencente à mesma classe. Isso resultaria em uma revocação perfeita, mas, claramente, seria um modelo inútil. Dessa forma, a revocação presta a demonstrar o quão bem determinado rótulo foi representado pelo modelo. Por fim, é possível concluir que, separadas, tais métricas podem apontar resultados aparentemente satisfatórios em casos de modelos falhos, mas unidas, elas representam melhor a realidade.

Ainda sobre métricas utilizadas para avaliação, a curva ROC (característica de operação do receptor - do inglês *receiver operating characteristic*) também foi levada em consideração. Esse gráfico é uma forma de mostrar como o desempenho de um classificador binário varia em função do limiar de decisão utilizado. Nesse trabalho, o limiar de decisão transforma a resposta da lógica *fuzzy* em uma classificação binária entre normalidade e anomalia. Assim, o limiar pode ser diminuído, para intensificar a sensibilidade do sistema a variações no tráfego, diminuindo sua tolerância; ou aumentado, para tornar o sistema mais tolerante.

A curva ROC é formada pela união de duas taxas: a de verdadeiros positivos e a de falsos positivos. Um limiar de decisão perfeito tem 100% de verdadeiros positivos e 0% de falsos positivos, contudo isso não é possível para todos os casos. Portanto, um limiar deve ser escolhido de forma que não negligencie nenhuma das duas taxas. Para isso, é escolhido o limiar que apresenta o maior valor de AUC (área sob a curva - do inglês *area under curve*), da curva ROC. O valor de AUC é a área calculada utilizando um dos pontos da curva ROC como vértice de um quadrilátero. Os valores podem variar de 0 (o pior) até 1 (o melhor) em função das taxas mencionadas. O melhor limiar de decisão é aquele que possuir o maior valor de AUC, já que apresenta o maior equilíbrio entre verdadeiros e falsos positivos.

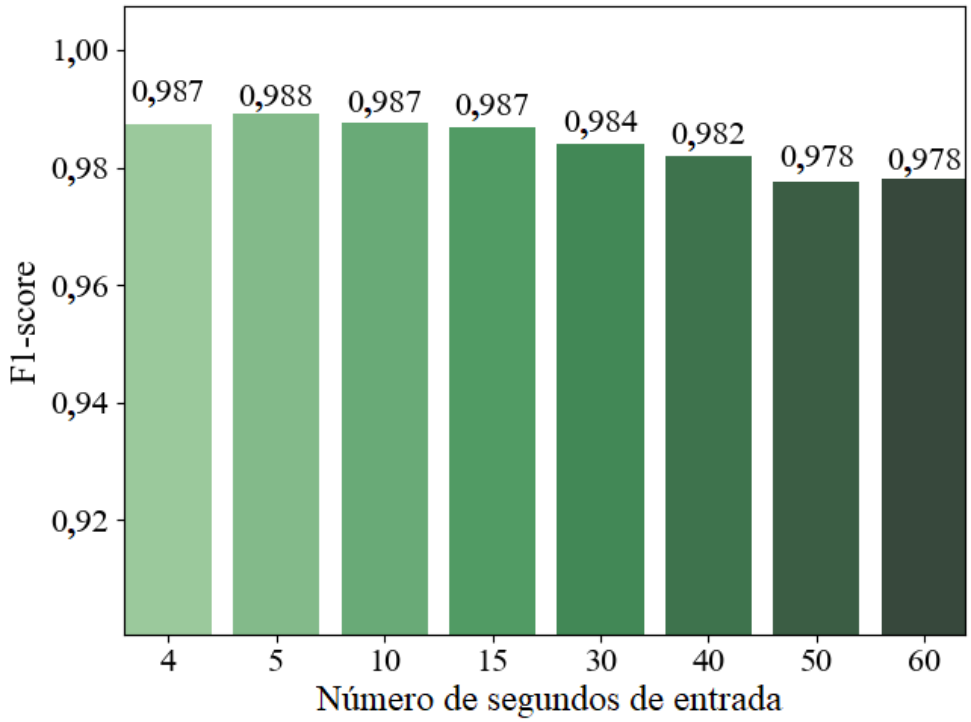

Figura 9 – Métrica *F1-Score* por quantidade de segundos de entrada

$$Precisão = \frac{Verdadeiro\ Positivo}{Verdadeiro\ Positivo + Falso\ Positivo} \tag{4.4}$$

$$Revocação = \frac{Verdadeiro\ Positivo}{Verdadeiro\ Positivo + Falso\ Negativo} \tag{4.5}$$

$$F1\text{-}score = 2\frac{Precisão \times Revocação}{Precisão + Revocação} \tag{4.6}$$

O primeiro hiperparâmetro ajustado foi o número de segundos de tráfego anteriores utilizados como entrada para a rede. A Figura 9 mostra os resultados de testes com 4, 5, 10, 15, 30, 40, 50 e 60 segundos de entrada. A melhor performance foi obtida com 5 segundos, o que demonstra que adicionar mais informação à rede neural na forma de segundos anteriores não melhora sua precisão. A diferença entre resultados não é tão grande, mesmo assim optou-se por utilizar 5 segundos, pois diminui o tempo de treinamento e evita adicionar informações desnecessárias à rede neural.

Em seguida, a quantidade de neurônios na camada recorrente da rede foi ajustada, e, como mostra a Figura 10, não houve uma diferença significativa entre os resultados.

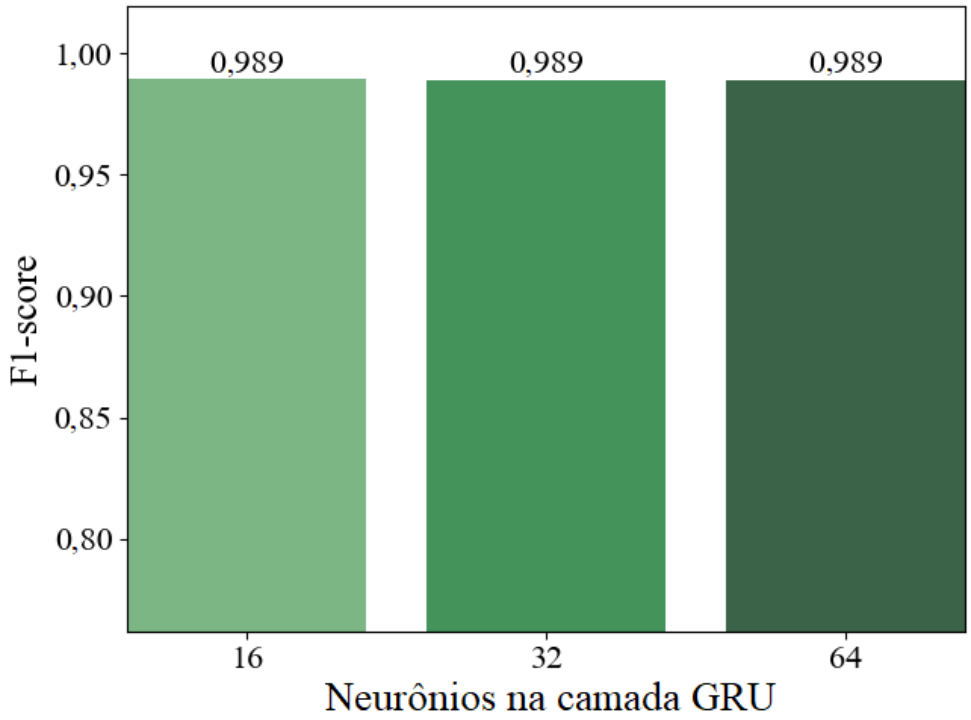

Figura 10 – Métrica *F1-Score* por quantidade de neurônios recorrentes

Além disso, também foram feitos testes com a quantidade de camadas densas não recorrentes. Seus resultados estão disponíveis na Figura 11 e mostram que o desempenho foi aproximado com uma e duas camadas, mas diminuiu com a terceira.

Para escolher um valor para quantidade de camadas e neurônios, foram avaliadas diferentes combinações de cada. Para todos os casos, a primeira camada sempre é composta por neurônios GRU, seguida por uma ou duas camadas densas tradicionais. Na Figura 12, estão apresentados os melhores resultados dentre os testes realizados. A quantidade de neurônios e camadas está representada como no exemplo do melhor teste: [32, 16, 1] significa 32 neurônios na camada recorrente, 16 na segunda camada e 1 na última.

Para o treinamento, o tamanho de *batch* foi avaliado com valores entre 120 e 420 inclusos em incrementos de 60. Os resultados são apresentados na Figura 13 e 360 foi escolhido por ter o melhor desempenho.

Por fim, a janela *fuzzy* foi ajustada. Ela representa a quantidade de segundos utilizados no cálculo de desvio padrão da função de pertinência *fuzzy*. A Figura 14 mostra os resultados de testes no intervalo de 5 a 60. O valor escolhido é 20, por ter o melhor desempenho com a menor janela e por evitar deixar o sistema mais tolerante do que o necessário.

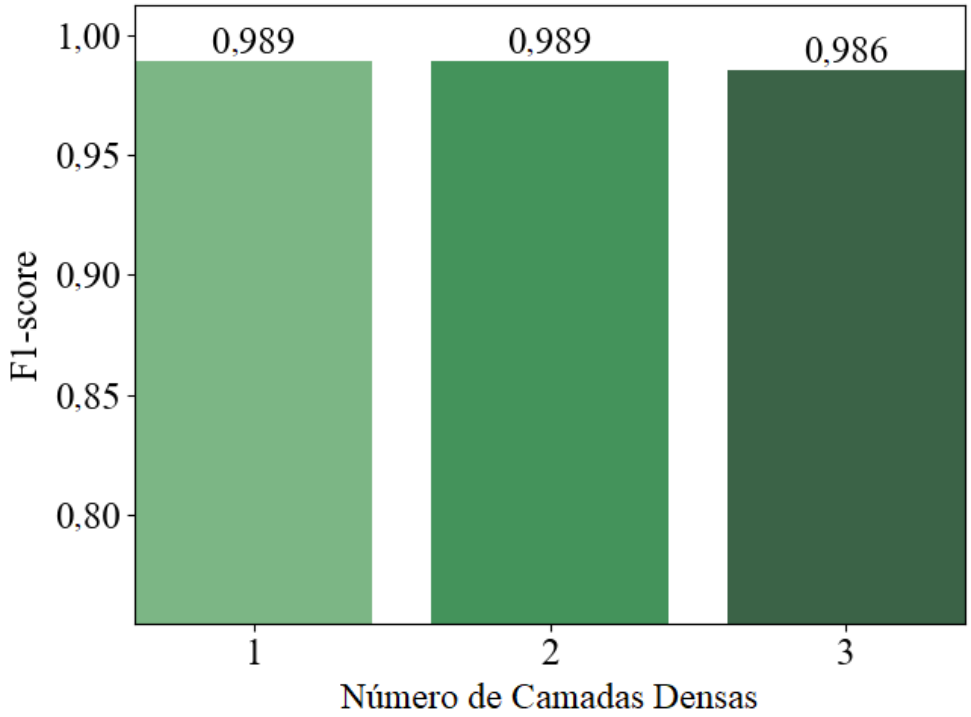

Figura 11 – Métrica *F1-Score* por quantidade de camadas densas não-recorrentes

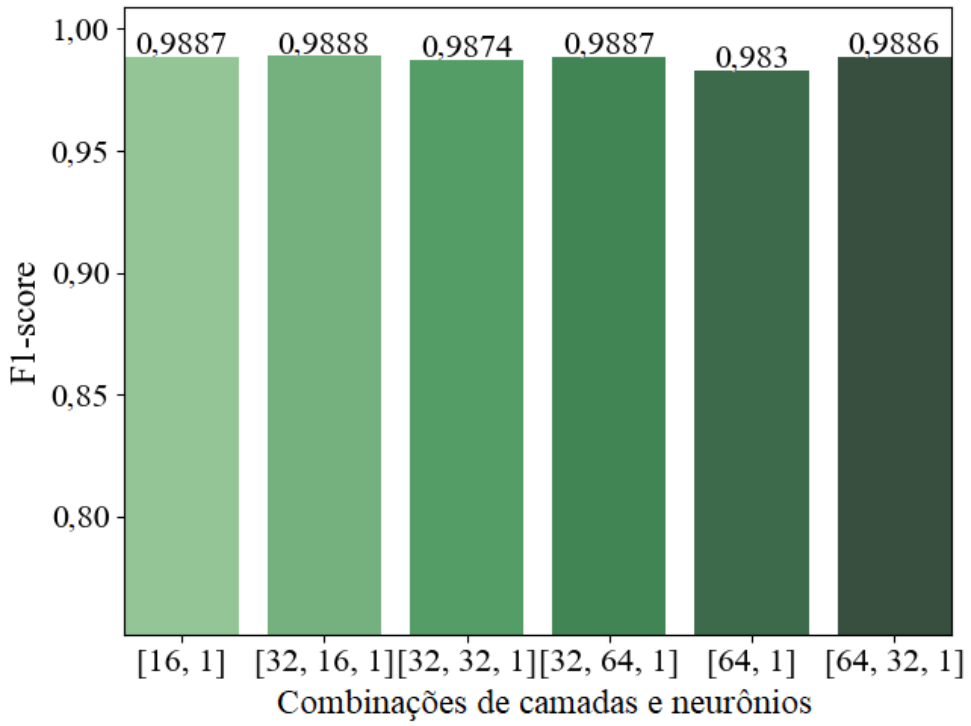

Figura 12 – Métrica *F1-Score* por combinação de neurônios

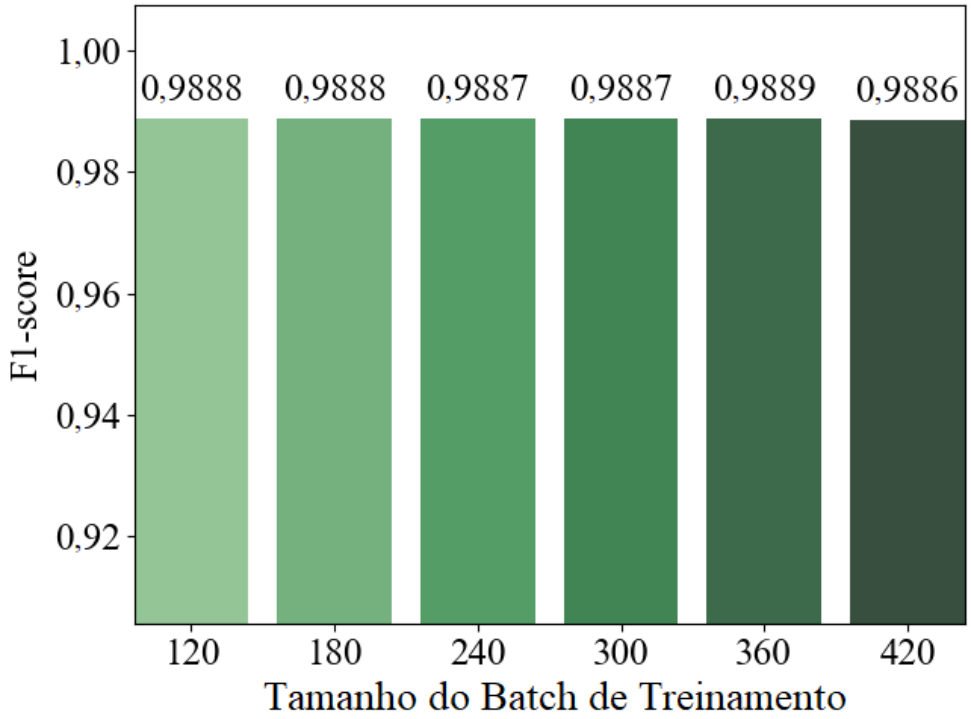

Figura 13 – Métrica *F1-Score* por tamanho do *batch*

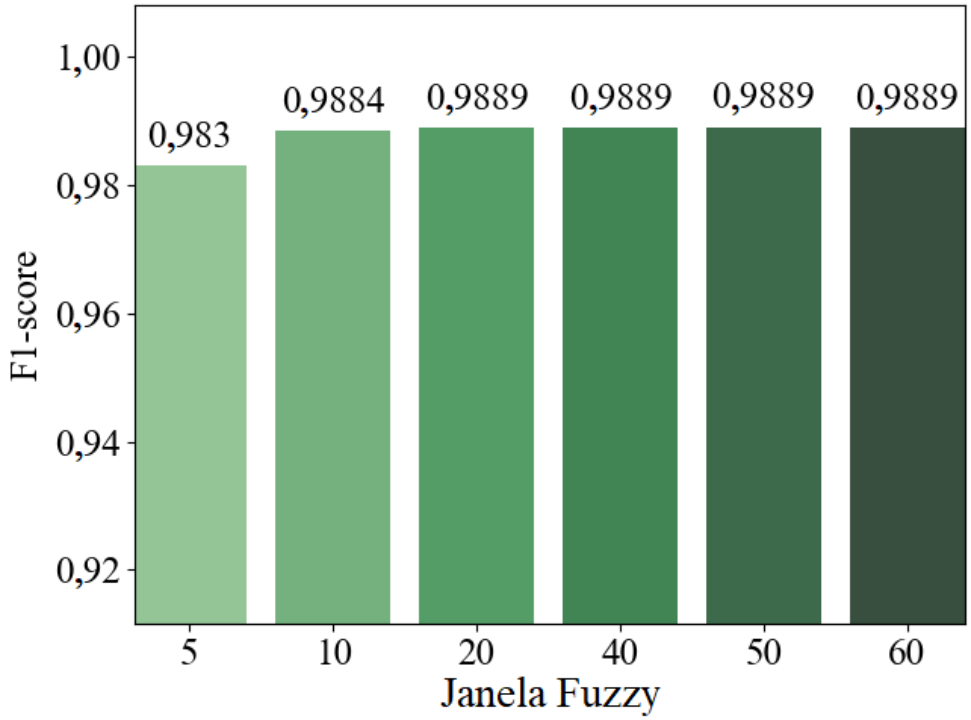

Figura 14 – Métrica *F1-Score* por janela *fuzzy*

5 ESTUDO DE CASO

5.1 Conjuntos de Dados

Para demonstrar a eficiência do modelo, um cenário foi utilizado. Os testes foram realizados em um uma máquina com *Windows* 10, processador *Ryzen* 1700x, 32GB de RAM e *Python* versão 3.8.7. Durante os testes, o objetivo foi maximizar a métrica *F1-score*, descrita na equação 4.6. O modelo proposto também foi testado com quatro outros tipos de redes neurais: a *long short-term memory*, a convolucional, a temporal convolucional e a tradicional.

5.1.1 Primeiro Conjunto

O primeiro conjunto de dados foi gerado pelo grupo Orion de pesquisa em redes de computadores da Universidade Estadual de Londrina e está disponível *online* em [78]. Para geração dos dados, um emulador de rede SDN chamado *mininet* [79] foi utilizado para criar uma topologia em árvore de seis *switches*, representada na figura 15. Nessa topologia, o primeiro *switch* é a raiz da árvore, enquanto os outros são as folhas conectadas a ele. Cada um dos cinco *switches* está conectado a doze hosts. O tráfego da rede foi gerado pela biblioteca de *Python Scapy* [80], que cria e envia pacotes pela rede. Para simular os ataques DDoS, o *software* hping3 [81] foi aplicado, enquanto ataques de *portscan* foram gerados pelo próprio *scapy*. Conjuntos similares já foram utilizados em outras publicações do grupo de redes [71] [82] [83].

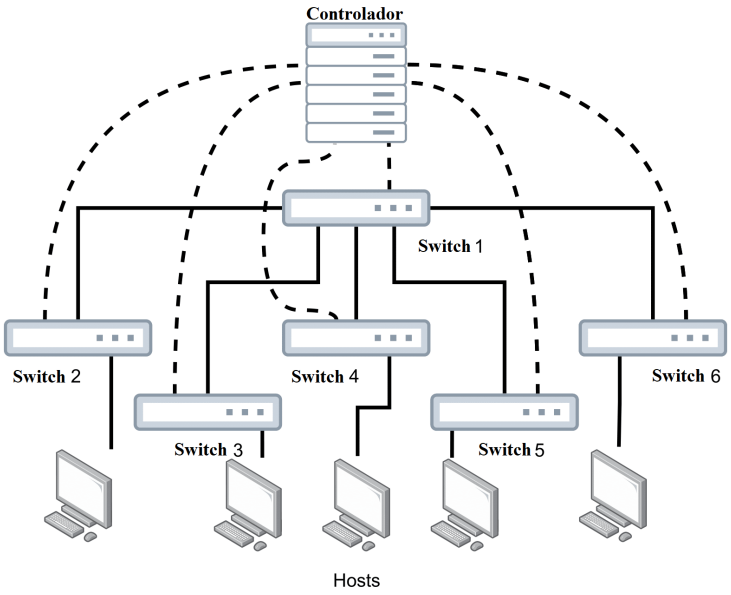

Figura 15 – Topologia da Rede do Cenário 1

O dataset contém 4 dias, sendo o primeiro deles sem ataque e os outros 3 com um ataque de DDoS e um *Portscan* em cada, sem que se sobreponham. Os dias têm duração de 24 horas e o dataset é organizado em fluxos. Por isso, é necessário fazer um pré-processamento dos dados para convertê-los nas dimensões de fluxo já citadas anteriormente. Durante a conversão, os dados são separados em segundos, em um total de 86400 segundos por dia. O dia sem ataques está representado na Figura 16, enquanto as Figuras 17, 18, e 19 apresentam os dias com ataques destacados em vermelho.

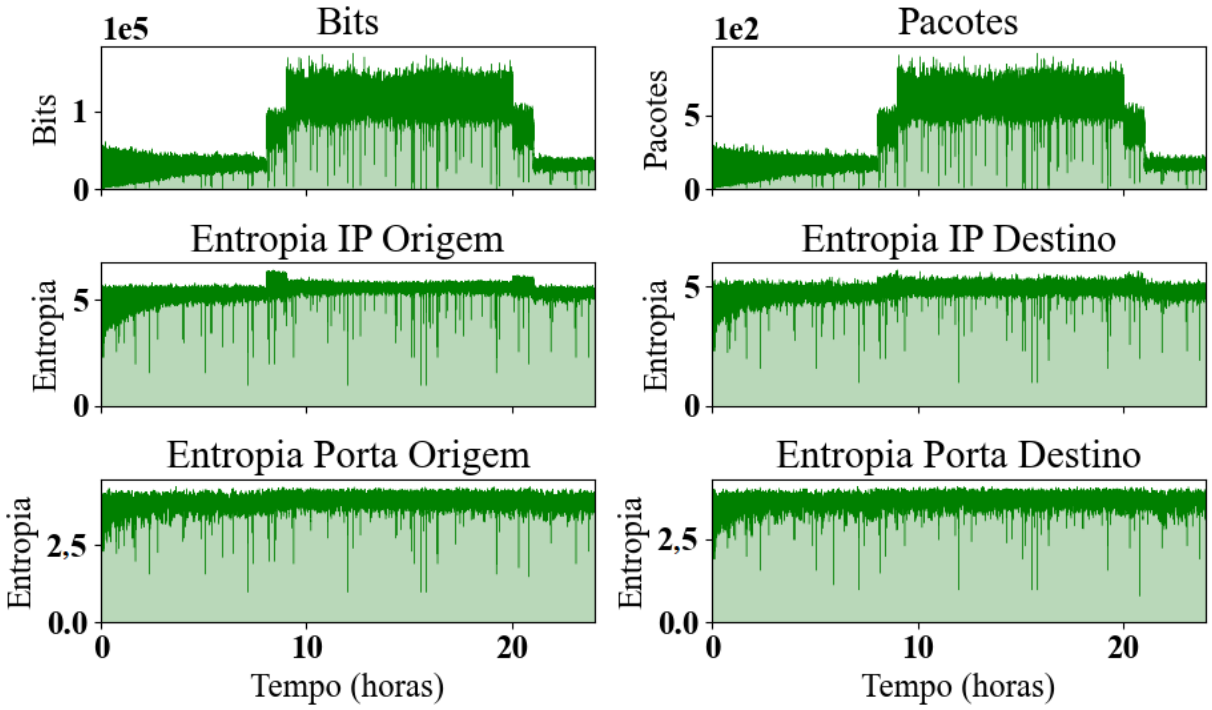

Figura 16 – Dia 1 sem ataque do primeiro conjunto de dados

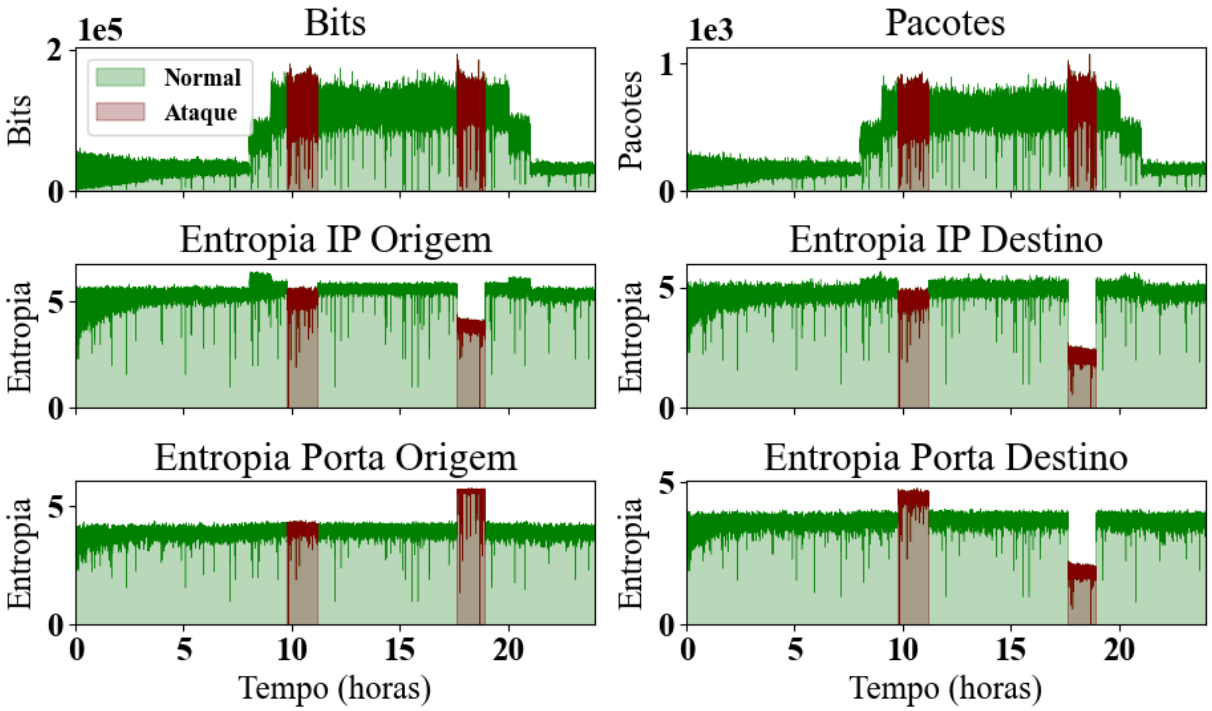

Figura 17 – Dia 2 com ataques do primeiro conjunto de dados

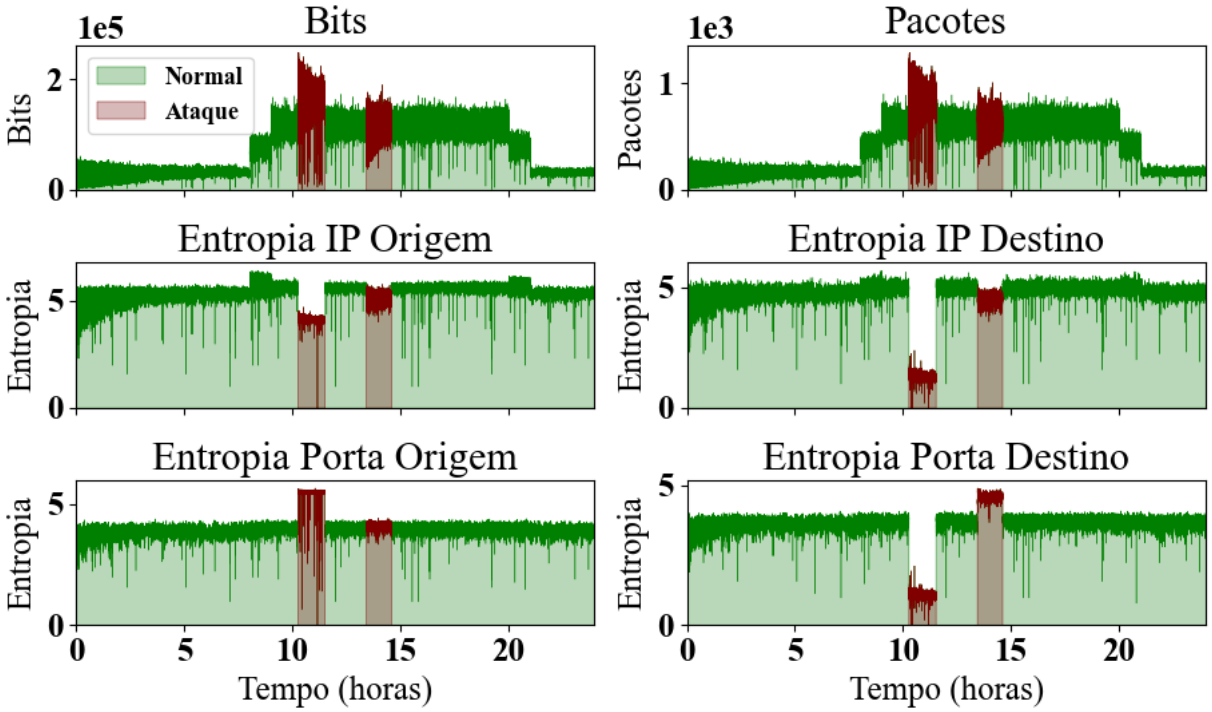

Figura 18 – Dia 3 com ataques do primeiro conjunto de dados

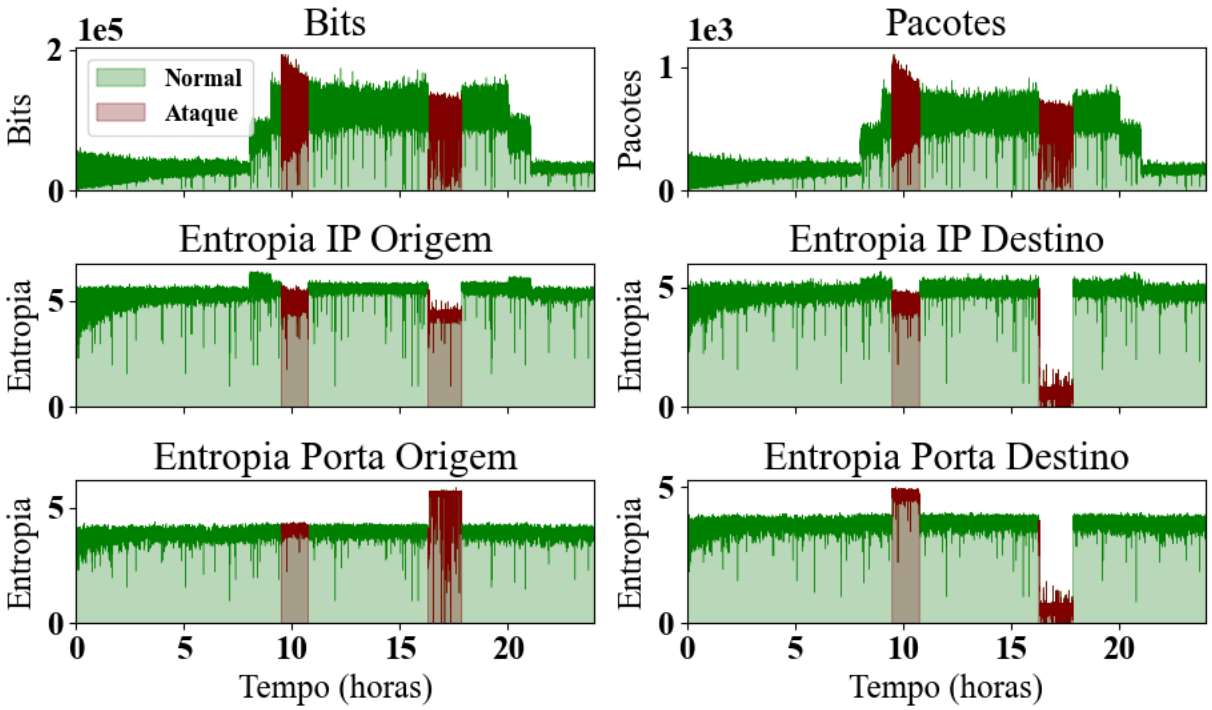

Figura 19 – Dia 4 com ataques do primeiro conjunto de dados

5.1.2 Segundo Conjunto

O segundo conjunto utilizado é chamado CICDDoS2019 e foi gerado e disponibilizado pela Universidade de Nova Brunswick [84]. O tráfego de dados de 25 usuários foi simulado por um sistema de perfilagem que procura se assemelhar ao comportamento humano. As anomalias são diferentes tipos de ataques de negação de serviço, que inclui: MSSQL, NetBios, SSDP, and UDP.

O conjunto tem 2 dias de dados, o primeiro com aproximadamente 4 horas de tráfego, e o segundo com apenas 2 horas. Após extrair as dimensões de fluxo e normalizá-las, o treinamento foi executado com o primeiro dia de dados sem os fluxos malignos (representado na figura 20). Em seguida, os testes utilizaram o segundo dia de dados, incluindo o tráfego anômalo representado na figura 21.

5.2 Testes e Resultados

5.2.1 Primeiro Cenário

Para o primeiro cenário, depois de treinar as redes neurais e calcular o limiar *fuzzy*, os testes foram feitos com o último dia de dados. Os resultados de detecção estão representados na matriz confusão da tabela 3 e na Figura 22. É possível observar que houve uma baixa taxa de falsos positivos e falsos negativos. A métrica *F1-score* resulta

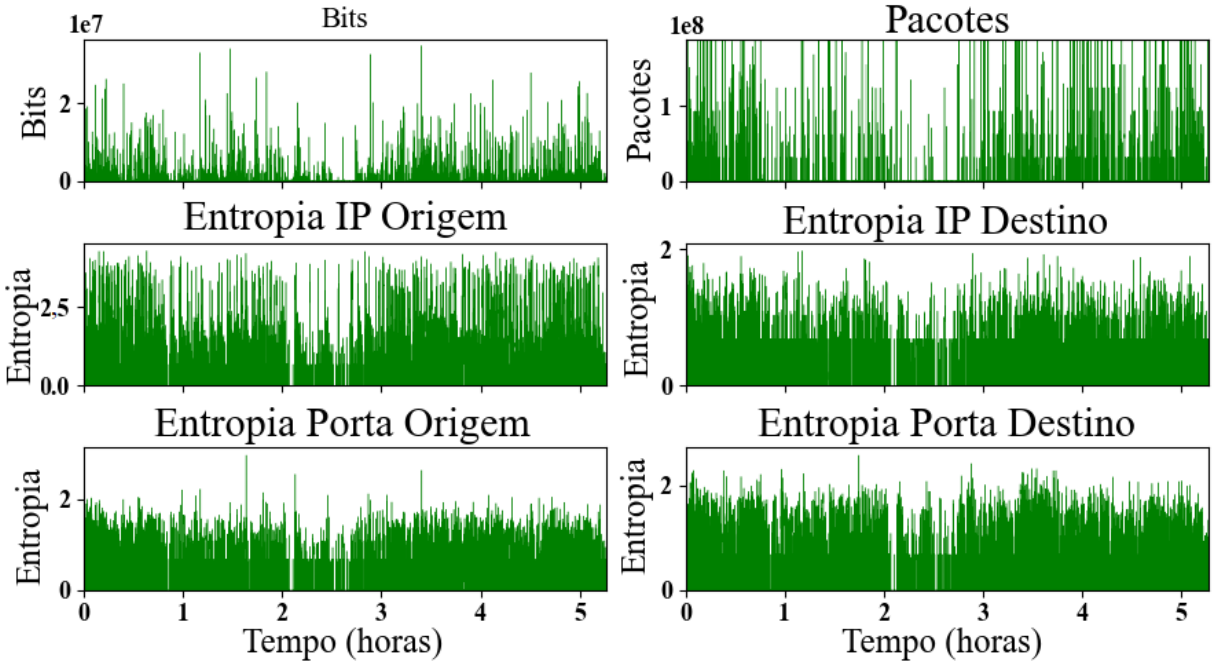

Figura 20 – Dia 1 sem ataque do segundo conjunto de dados

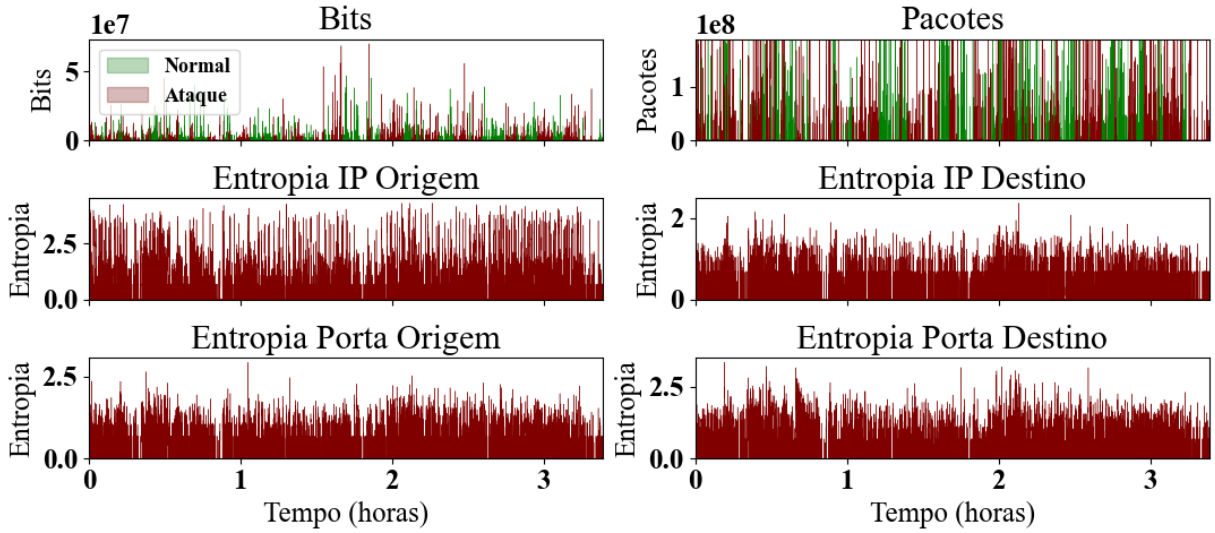

Figura 21 – Dia 2 com ataque do segundo conjunto de dados

Tabela 3 – Matriz Confusão de Detecção do Primeiro Cenário

	Nenhum Ataque Detectado	Ataque Detectado
Rótulo: Sem Ataque	76.236	20
Rótulo: Com Ataque	191	9.949

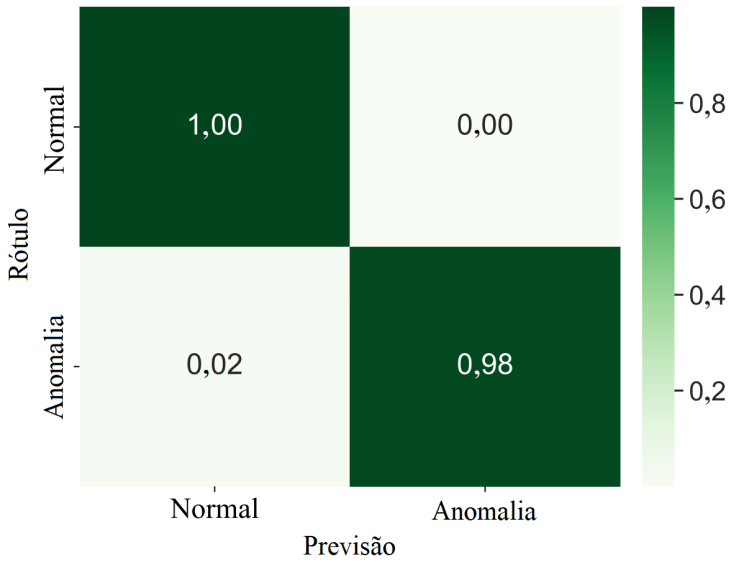

Figura 22 – Matriz Confusão Normalizada do Primeiro Cenário

em 98.9%, e a precisão e revocação são 99.7% e 98.1%, respectivamente, como apresentado na Figura 23.

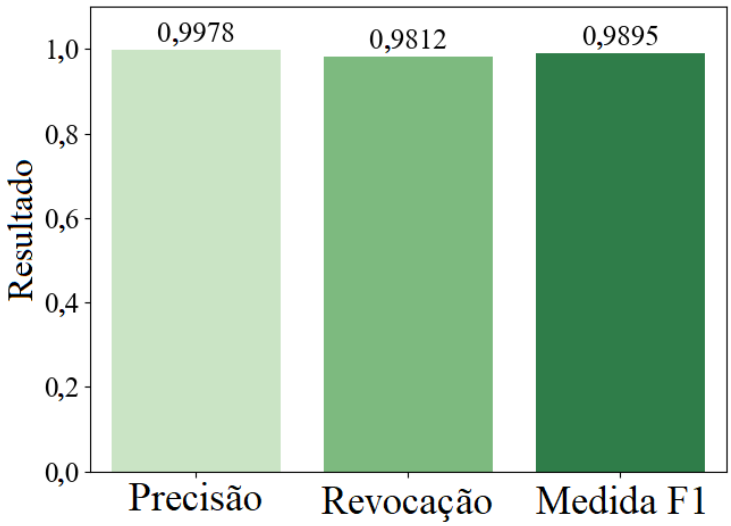

Figura 23 – Métricas do Conjunto de Testes do Primeiro Cenário

Ao comparar o resultado da detecção dos outros modelos de rede neural na tabela 4, é possível notar que a rede tradicional (DNN) é significativamente pior que as outras que

obtiveram resultados semelhantes, com menos de 1% de diferença entre si, mas com uma leve vantagem para a TCN na métrica *F1-score*. Devido a variações no comportamento da rede, é comum a presença de "*outliers*" (entradas que divergem do comportamento normal mas não são maliciosas), por isso é inevitável que erros de classificação sejam cometidos. Suas consequências seriam de bloqueio de fluxos inocentes e liberação de fluxos maliciosos, mas a lista segura e a lista de bloqueios têm como função evitar que aconteçam durante a mitigação.

A figura 24 mostra a curva ROC da escolha de limiar da lógica fuzzy. As áreas do melhor limiar de cada curva são: 0,93, 0,99, 0,99, 0,97, e 0,98, para os modelos DNN, CNN, LSTM, GRU, e TCN, respectivamente. É possível notar que o desempenho na curva roc não reflete perfeitamente nas métricas de teste, já que a LSTM e a CNN tiveram as melhores áreas, mas não as melhores métricas F1; mas ainda são um indicativo de desempenho. Essa pequena variação acontece devido a como cada rede se adapta ao conjunto de treino.

Tabela 4 – Comparação de Resultados Cenário 1

	Precisão	Revocação	F1-score
DNN	0,9507	0,8616	0,9040
CNN	0,9745	**0,9940**	0,9843
LSTM	0,9829	0,9829	0,9829
GRU	**0,9978**	0,9812	0,9895
TCN	0,9943	0,9877	**0,9910**

Sobre a mitigação, a tabela 5 representa a matriz confusão. É possível observar que, mesmo com alguns falsos positivos durante a detecção, foram raros os casos de fluxos bloqueados injustamente. Isso se deve à implementação da lista segura, que dificulta que usuários legítimos sejam bloqueados. A Figura 25 ilustra a diferença de fluxos antes e depois do algoritmo de mitigação.

É importante ressaltar que ataques *portscan* são mais difíceis de detectar do que ataques DDoS por não perturbarem tão significativamente as dimensões de fluxo. Isso acontece porque o objetivo do ataque não é sobrecarregar o sistema, mas sim coletar

Tabela 5 – Resultados Mitigação do Primeiro Cenário

	Total Mitigados	Total Aceitos
Fluxos Maliciosos	2.959.021 (99,99%)	18 (0,01%)
Fluxos Normais	264 (0,01%)	3.957.094 (99,99%)

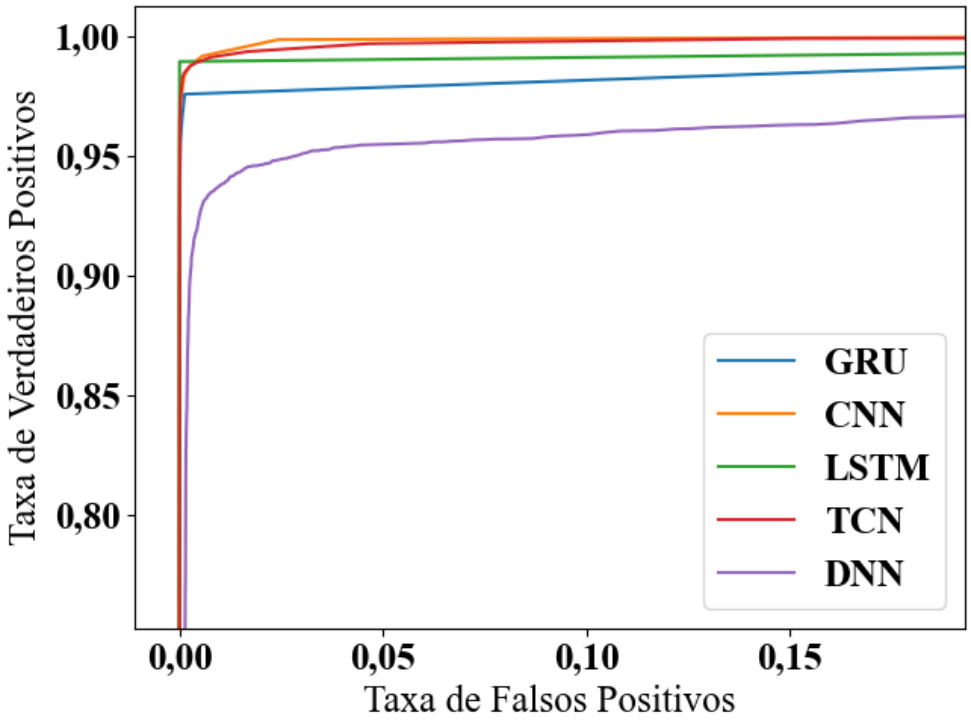

Figura 24 – Curva ROC do treinamento do primeiro cenário

informações. O grau de relevância do ataque para as *features* utilizadas depende de quantos fluxos são maliciosos em relação à quantidade de fluxos normais, de forma que é possível que o ataque não seja detectado por não se destacar sobre os demais fluxos.

5.3 Segundo Cenário

O primeiro dia de dados foi dividido para treinamento, validação e ajuste de limiar, o que possibilitou que todo o segundo dia de dados fosse utilizado para testes. Os resultados da detecção estão apresentados na tabela 6 e na figura 26. Por ser um cenário de dados com menos horas de tráfego, a quantidade de entradas no conjunto de teste é reduzida em relação ao outro cenário. A reduzida quantidade de dados também afeta o desempenho do modelo, que normalmente se beneficia de grandes quantidades de entradas, por se tratar de aprendizagem profunda. Contudo, mesmo com uma quantidade limitada de dados, foi possível alcançar resultados acima de 90%.

Tabela 6 – Resultados de detecção do segundo cenário

	Nenhum Ataque Detectado	Ataque Detectado
Rótulo: sem ataque	181	116
Rótulo: com ataque	50	1.376

A tabela 6 e a Figura 26 representam a matriz confusão do teste, enquanto a Figura

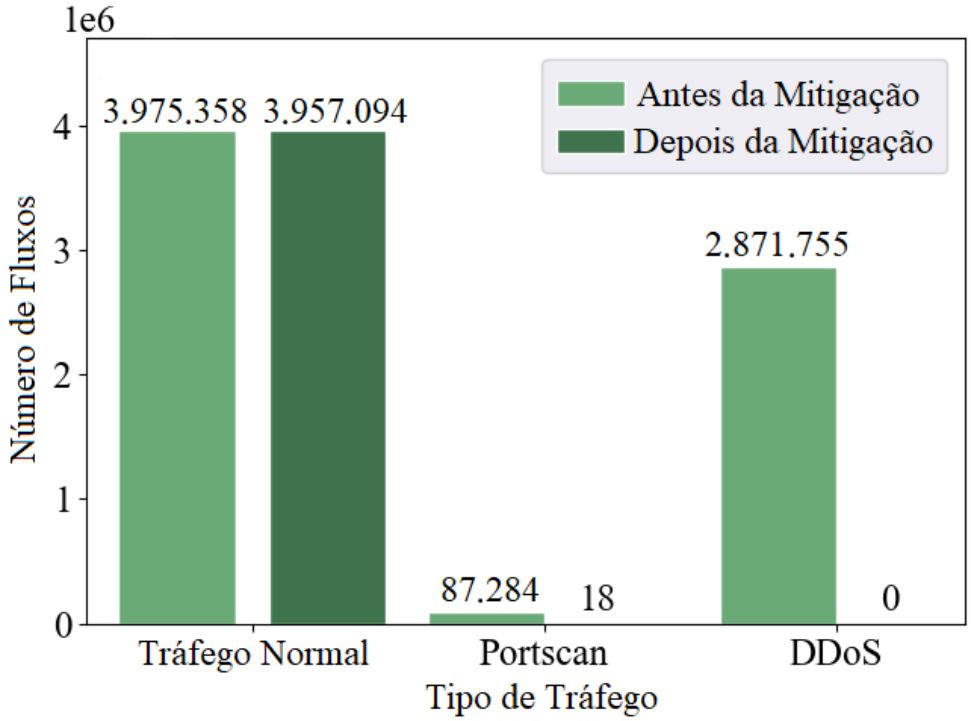

Figura 25 – Quantidade de Fluxos Antes e Depois da Mitigação no Primeiro Cenário

27 mostra as métricas resultantes. Os resultados da comparação com outros tipos de redes neurais estão retratados na Tabela 7. Aqui também é possível notar um desempenho menor da rede neural tradicional em relação às outras que apresentam resultados similares entre si. Mais uma vez, a GRU consegue uma precisão maior, enquanto a TCN tem um *F1-score* melhor, ainda que por muito pouco. A precisão de um modelo diminui quando falsos positivos acontecem, consequentemente, é possível dizer que, quanto melhor a precisão de um modelo, melhor este conseguiu aprender e caracterizar o tráfego normal da rede, mantendo um menor erro durante a previsão na ausência de anomalias. Por isso, é possível dizer que as redes GRU conseguiram uma melhor caracterização do tráfego, enquanto as redes TCN alcançaram um melhor desempenho na detecção de anomalias em ambos os cenários, mesmo que por uma margem pequena.

A curva ROC apresentada na figura 28 mostra uma variação maior nas taxas de verdadeiros e falsos positivos em relação ao primeiro cenário. Isso ocorre principalmente por dois motivos: a já mencionada escassez de dados para treinamento e teste, e a maior variação do tráfego da rede desse conjunto em relação ao segundo. É notável como na figura 20 as dimensões de fluxo seguem um padrão menos uniforme do que o observado nas figuras 17, 18, e 19. Isso cria uma dificuldade maior para a rede neural se adaptar aos dados, o que requer que a tolerância do limiar de decisão seja maior, que por sua vez gera mais falsos negativos, diminuindo o desempenho do modelo.

Para a mitigação, o mesmo algoritmo foi aplicado e o resultado está apresentado na Figura 29 e na Tabela 8. Nota-se que a maior parte dos fluxos maliciosos foi descartada,

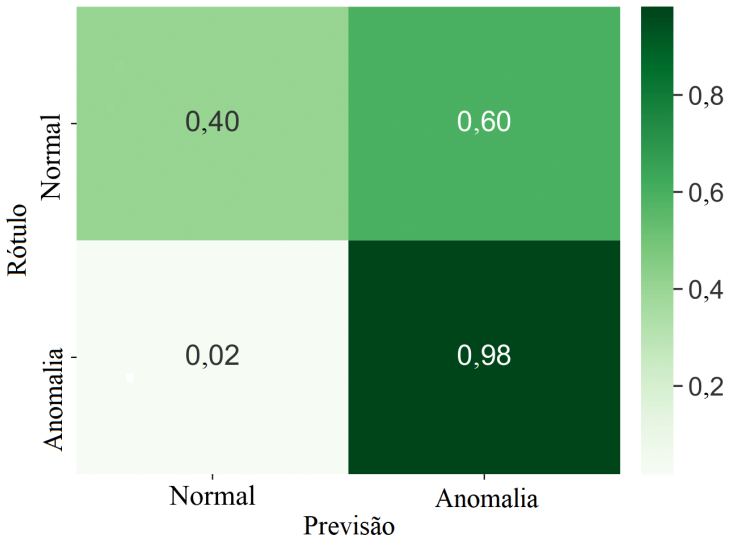

Figura 26 – Matriz Confusão Normalizada do Segundo Cenário

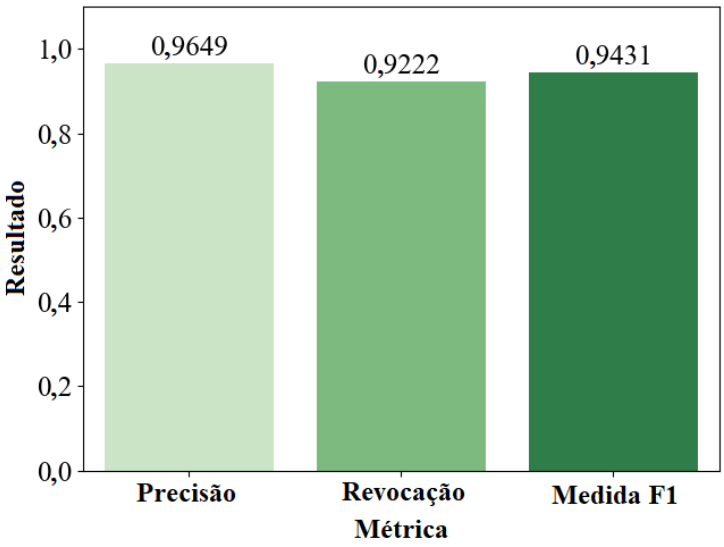

Figura 27 – Métricas do Conjunto de Testes do Segundo Cenário

Tabela 7 – Comparação de Resultados Cenário 2

	Precisão	Revocação	F1-score
DNN	0,8534	0,9739	0,9097
CNN	0,8818	**0,9894**	0,9324
LSTM	0,8859	0,9859	0,9332
GRU	**0,9222**	0,9649	0,9431
TCN	0,9214	0,9789	**0,9493**

mas houve um aumento nos falsos positivos, mesmo com a lista segura evitando bloqueios injustos. É provável que diferentes tipos de rede precisem de ajustes na quantidade de

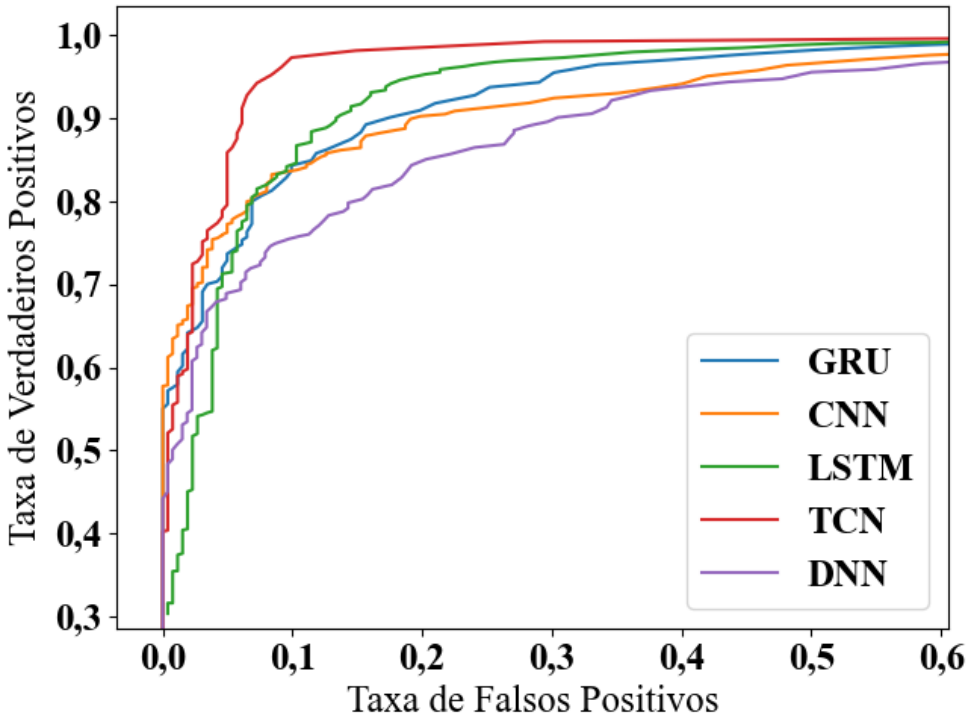

Figura 28 – Curva ROC do Treinamento do Segundo Cenário

segundos de imunidade em uma lista segura, baseado no comportamento médio de usuários. Talvez uma lista segura com uma quantidade variável de segundos que se baseasse no comportamento de usuários em tempo real poderia ser objeto de estudo de um trabalho futuro. É claro que, nesse caso, a qualidade da detecção também influenciou o aumento relativo da quantidade de falsos positivos da mitigação em relação ao primeiro cenário.

Tabela 8 – Resultados Mitigação do Segundo Cenário

	Total Mitigado	Total Aceito
Fluxos Maliciosos	20.930 (96,45%)	769 (3,54%)
Fluxos Normais	1626 (22,43%)	5.620 (77,56%)

5.4 Considerações Sobre os Cenários

É possível notar como o primeiro cenário tem um desempenho superior ao segundo. Isso pode ser explicado por dois fatores principais: a quantidade de dados disponível para treinamento no primeiro cenário é maior; e o comportamento de sua rede é mais "organizado" que o segundo cenário, o que facilita a adaptação dos modelos. Separadamente, os fatores mencionados não afetariam tanto o desempenho do sistema de detecção, mas o problema é agravado quando ambos ocorrem ao mesmo tempo.

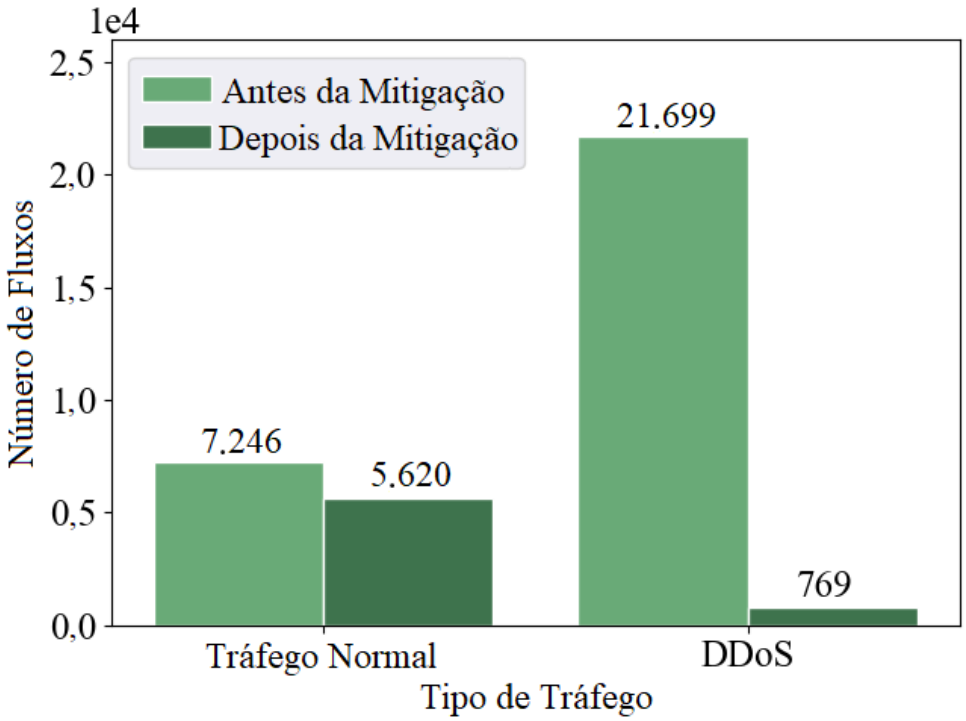

Figura 29 – Quantidade de Fluxos Antes e Depois da Mitigação no Segundo Cenário

Mesmo com muitas variações no comportamento da rede, dados suficientes podem possibilitar que uma rede neural encontre padrões. Em um ambiente real, seria possível continuamente coletar dados para aumentar o conjunto de treinamento e melhorar o sistema de detecção de anomalias, tornando o problema de falta de dados apenas uma questão de tempo.

A mitigação em ambos os cenários é satisfatória, pois os fluxos maliciosos foram bloqueados até o ponto de que não causariam dano significativo à rede. Mesmo com diferentes tipos de ataques, o sistema baseado em anomalia possibilita que apenas com o aprendizado do comportamento normal seja possível detectar anomalias ao apontar variações significativas no tráfego.

5.5 Custo Computacional

Durante o desenvolvimento de um *software*, é importante ter em mente sua complexidade para que sua viabilidade não seja comprometida. Para isso, o processo de análise de algoritmo calcula como o tempo de execução de um programa aumenta em função do tamanho de sua entrada. Esse processo também é chamado de análise assintótica.

Neste trabalho, será estudada a notação de O maiúsculo (*big O notation*). Essa análise usa o pior caso do algoritmo para calcular sua performance enquanto ignora casos menos complexos ou de menor grau de complexidade. Apesar de não ser uma análise perfeita do modo como o algoritmo se comportará no mundo real, é um bom indicativo

para guiar seu desenvolvimento. A notação é representada pela letra “O”, seguida por um indicativo, em parênteses, da complexidade do algoritmo em função de uma entrada de tamanho N. Por exemplo, um algoritmo $O(N^2)$ tem um crescimento do tempo de execução exponencial ao crescimento da entrada

Para analisar todo o algoritmo, ele será dividido em partes, seguindo a descrição do sistema proposto no capítulo 4. A primeira etapa considerada é o processamento dos fluxos, para que sejam extraídas as dimensões de fluxo utilizadas pela rede neural. O processo de cálculo de bits e pacotes por segundo é apenas uma soma, por isso é O(N). Já o processo de cálculo de entropias percorre os dados três vezes, ou menos, para cada *feature*, resultando em quatro vezes O(3N). Por isso, pode-se concluir que o pré-processamento dos dados tem complexidade de O(N).

Seguindo o sistema proposto, o próximo passo é a previsão das redes neurais. Essa etapa recebe os dados já processados no trecho anterior, por isso, a entrada tem tamanho fixo e o tempo de processamento é o mesmo, independentemente de quantos fluxos percorreram a rede no último segundo. A execução de uma rede neural não é custosa depois que seu treinamento foi finalizado. Durante os testes, a rede neural precisou de 3,5 segundos para processar os dados de 24 horas de tráfego, o que mostra que seu impacto no processamento é insignificante.

A previsão da rede neural é combinada com o valor real esperado das dimensões de fluxo na função de pertinência da lógica difusa. Esse processo também tem uma entrada de tamanho fixo e, consequentemente, tempo constante de processamento.

Por fim, o algoritmo de mitigação tem dois casos principais: os ataques de *portscan* e os de DDoS. Para o caso de um *portscan*, o algoritmo precisa percorrer todos os fluxos no máximo duas vezes, o que torna o processo O(2N). Um ataque DDoS requer que os fluxos sejam percorridos até 3 vezes, tornando o processo O(3N). As listas seguras podem ser implementadas utilizando dicionários ou tabelas *hash*, o que torna o processo de busca N vezes O(1). Por isso, é possível concluir que o algoritmo de mitigação é O(N).

6 CONCLUSÃO E TRABALHOS FUTUROS

Detectar e mitigar anomalias no tráfego é fundamental para o bom funcionamento de uma rede. Este trabalho apresenta um sistema de detecção de anomalias que utiliza seis redes neurais GRU para prever seis dimensões de fluxo que representam o tráfego de rede. Os modelos só são treinados utilizando tráfego normal, por isso é seguro dizer que a rede neural fará previsões menos precisas quando dados fora da normalidade estiverem presentes. Utilizando a previsão das redes neurais, uma função de pertinência irá decidir se o tráfego real possui uma anomalia ou não. A tolerância dessa função é ajustada utilizando um dia com ataques, o que torna esse sistema um modelo supervisionado.

Em ambos os cenários, as redes GRU, CNN, LSTM e TCN apresentaram desempenho similar entre si. A rede neural tradicional não apresentou resultados satisfatórios e ficou abaixo das outras. Isso mostra a importância da representação da ordem do tempo numa previsão de tráfego de rede. Os modelos que possuem alguma forma de organização de contexto (seja por memória no caso da LSTM e na GRU, ou por representação espacial no caso da CNN e TCN) alcançaram um maior desempenho em relação às redes tradicionais.

No primeiro cenário, é possível notar como ataques de DDoS são mais volumosos e perturbam mais a rede do que ataques de *portscan*. Isso torna os DDoS mais danosos a curto prazo, mas, ao mesmo tempo, são mais simples de detectar. Os ataques *portscan* tendem a ser mais difíceis de identificar por terem um menor número de fluxos e por seu propósito ser de apenas coletar informações, em contraste com o DDoS que busca interromper o funcionamento do sistema. Mesmo assim, o sistema proposto conseguiu detectar e mitigar a maior parte dos fluxos atacantes de ambas as anomalias.

No segundo cenário, os modelos mais uma vez alcançaram resultados semelhantes, com exceção da rede neural tradicional. Esse *dataset* tem como maior obstáculo a baixa quantidade de horas de tráfego para treino e teste das redes neurais. Já é sabido que modelos mais profundos se beneficiam de maiores volumes de dados, por isso é interessante saber que, mesmo com uma quantidade menor de segundos de tráfego, o modelo conseguiu uma performance acima de 93%. Mesmo assim, a quantidade de falsos positivos foi maior, o que mostra uma dificuldade na caracterização do tráfego normal da rede, provavelmente oriundo dessa carência de dados.

Em relação à mitigação dos ataques, é evidente que as informações confiáveis de detecção são essenciais para seu bom funcionamento. Contudo, a implementação da lista segura possibilita que falsos positivos resultem em menos fluxos inocentes sendo descartados, assim como a lista de bloqueio evita que fluxos malignos sejam ignorados na ocorrên-

cia de falsos negativos. Como mencionado, diferentes perfis de tráfego e usuários podem precisar de diferentes durações de listas seguras e de bloqueio. Por isso, um algoritmo de ajuste de tempo das listas pode ser útil para aumentar sua eficiência.

Para trabalhos futuros, planeja-se desenvolver um sistema baseado em anomalia que não necessite de dados rotulados para ser treinado. Dessa forma, o sistema estaria ainda mais sensível a anomalias desconhecidas. Uma possível abordagem para um sistema como esse é a utilização de redes generativas adversárias. Esse tipo de rede utiliza duas redes neurais (geradora e discriminadora) que competem entre si para melhorar em suas respectivas funções: gerar tráfego falso semelhante ao verdadeiro, e diferenciar tráfego verdadeiro do falso. Dessa forma, um discriminador treinado apenas com tráfego real pode detectar tráfego anômalo, sem a necessidade de calibração de sensibilidade.

Além disso, é importante estudar novas bases de dados para validar sistemas de detecção de anomalias. Planeja-se utilizar bases de dados reais, com maior volume de dados. Um exemplo é a base UGR-16, coletada por um provedor de internet espanhol com fluxos rotulados e com ataques, tanto reais (detectados pelo próprio provedor) quanto ataques sintéticos [85].

REFERÊNCIAS

[1] AL-RAHAMNEH, A.; ASTRAIN, J. J.; VILLADANGOS, J.; KLAINA, H.; GUEMBE, I. P.; LOPEZ-ITURRI, P.; FALCONE, F. Enabling customizable services for multimodal smart mobility with city-platforms. *IEEE Access*, v. 9, p. 41628–41646, 2021.

[2] MYLONAS, G.; KALOGERAS, A.; KALOGERAS, G.; ANAGNOSTOPOULOS, C.; ALEXAKOS, C.; MUñOZ, L. Digital twins from smart manufacturing to smart cities: A survey. *IEEE Access*, v. 9, p. 143222–143249, 2021.

[3] MANGLA, C.; RANI, S.; Faseeh Qureshi, N. M.; SINGH, A. Mitigating 5g security challenges for next-gen industry using quantum computing. *Journal of King Saud University - Computer and Information Sciences*, 2022. ISSN 1319-1578. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1319157822002373>.

[4] WI-FI Alliance. Https://www.wi-fi.org. Acessado pela última vez: 04-05-2022.

[5] LI, C.; JIANG, K.; LUO, Y. Dynamic placement of multiple controllers based on sdn and allocation of computational resources based on heuristic ant colony algorithm. *Knowledge-Based Systems*, v. 241, p. 108330, 2022. ISSN 0950-7051. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950705122001204>.

[6] ALHIJAWI, B.; ALMAJALI, S.; ELGALA, H.; Bany Salameh, H.; AYYASH, M. A survey on dos/ddos mitigation techniques in sdns: Classification, comparison, solutions, testing tools and datasets. *Computers and Electrical Engineering*, v. 99, p. 107706, 2022. ISSN 0045-7906. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0045790622000234>.

[7] BOUR, H.; ABOLHASAN, M.; JAFARIZADEH, S.; LIPMAN, J.; MAKHDOOM, I. A multi-layered intrusion detection system for software defined networking. *Computers and Electrical Engineering*, v. 101, p. 108042, 2022. ISSN 0045-7906. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0045790622003032>.

[8] JIMéNEZ, M. B.; FERNáNDEZ, D.; RIVADENEIRA, J. E.; BELLIDO, L.; CáRDENAS, A. A survey of the main security issues and solutions for the sdn architecture. *IEEE Access*, v. 9, p. 122016–122038, 2021.

[9] FOULADI, R. F.; ERMI, 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://www.sciencedirect.com/science/article/pii/S1389128622002560>.

[10] VARGHESE, J. E.; MUNIYAL, B. Trend in sdn architecture for ddos detection-a comparative study. In: *2021 IEEE International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)*. [S.l.: s.n.], 2021. p. 170–174.

[11] AHALAWAT, A.; BABU, K. S.; TURUK, A. K.; PATEL, S. A low-rate ddos detection and mitigation for sdn using renyi entropy with packet drop. *Journal of Information Security and Applications*, v. 68, p. 103212, 2022. ISSN 2214-2126. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2214212622000898>.

[12] GONZáLEZ-MUñIZ, A.; DíAZ, I.; CUADRADO, A. A.; GARCíA-PéREZ, D.; PéREZ, D. Two-step residual-error based approach for anomaly detection in engineering systems using variational autoencoders. *Computers and Electrical Engineering*, v. 101, p. 108065, 2022. ISSN 0045-7906. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0045790622003226>.

[13] XIAOLAN, W.; Manjur Ahmed, M.; Nizam Husen, M.; QIAN, Z.; BELHAOUARI, S. B. Evolving anomaly detection for network streaming data. *Information Sciences*, v. 608, p. 757–777, 2022. ISSN 0020-0255. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0020025522006582>.

[14] POSILOVI, L.; MEDAK, D.; MILKOVI, F.; SUBAI, M.; BUDIMIR, M.; LONARI, S. Deep learning-based anomaly detection from ultrasonic images. *Ultrasonics*, v. 124, p. 106737, 2022. ISSN 0041-624X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0041624X2200049X>.

[15] de Paula Monteiro, R.; LOZADA, M. C.; MENDIETA, D. R. C.; LOJA, R. V. S.; FILHO, C. J. A. B. A hybrid prototype selection-based deep learning approach for anomaly detection in industrial machines. *Expert Systems with Applications*, v. 204, p. 117528, 2022. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S095741742200851X>.

[16] LIU, T.; WANG, H.; ZHANG, Y. A traffic anomaly detection scheme for non-directional denial of service attacks in software-defined optical network. *Computers Security*, v. 112, p. 102467, 2022. ISSN 0167-4048. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167404821002911>.

[17] TAO, X.; PENG, Y.; ZHAO, F.; YANG, C.; QIANG, B.; WANG, Y.; XIONG, Z. Gated recurrent unit-based parallel network traffic anomaly detection using subagging ensembles. *Ad Hoc Networks*, v. 116, p. 102465, 2021. ISSN 1570-8705. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1570870521000342>.

[18] RADAIDEH, M. I.; PAPPAS, C.; WALDEN, J.; LU, D.; VIDYARATNE, L.; BRITTON, T.; RAJPUT, K.; SCHRAM, M.; COUSINEAU, S. Time series anomaly detection in power electronics signals with recurrent and convlstm autoencoders. *Digital Signal Processing*, p. 103704, 2022. ISSN 1051-2004. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1051200422003219>.

[19] KHAN, N.; ELIZONDO, D. A.; DEKA, L.; MOLINA-CABELLO, M. A. Fuzzy logic applied to system monitors. *IEEE Access*, v. 9, p. 56523–56538, 2021.

[20] BATCHU, R. K.; SEETHA, H. An integrated approach explaining the detection of distributed denial of service attacks. *Computer Networks*, v. 216, p. 109269, 2022. ISSN 1389-1286. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1389128622003334>.

[21] RANI, S. J.; IOANNOU, I.; NAGARADJANE, P.; CHRISTOPHOROU, C.; VASSILIOU, V.; CHARAN, S.; PRAKASH, S.; PAREKH, N.; PITSILLIDES, A. Detection of ddos attacks in d2d communications using machine learning approach. *Computer Communications*, v. 198, p. 32–51, 2023. ISSN 0140-3664. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0140366422004364>.

[22] AYDN, H.; ORMAN, Z.; AYDN, M. A. A long short-term memory (lstm)-based distributed denial of service (ddos) detection and defense system design in public cloud network environment. *Computers Security*, v. 118, p. 102725, 2022. ISSN 0167-4048. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167404822001201>.

[23] YAZDINEJAD, A.; DEHGHANTANHA, A.; KARIMIPOUR, H.; SRIVASTAVA, G.; PARIZI, R. M. An efficient packet parser architecture for software-defined 5g networks. *Physical Communication*, v. 53, p. 101677, 2022. ISSN 1874-4907. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1874490722000453>.

[24] BAZIANA, P.; DRAINAKIS, G.; SYKAS, E. Software-defined optical intra-data center network and access control strategy. *Optical Switching and Networking*, v. 45, p. 100679, 2022. ISSN 1573-4277. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1573427722000157>.

[25] ARYAN, R.; YAZIDI, A.; BRATTENSBORG, F.; KURE Øivind; ENGELSTAD, P. E. Sdn spotlight: A real-time openflow troubleshooting framework. *Future Generation Computer Systems*, v. 133, p. 364–377, 2022. ISSN 0167-739X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167739X22000899>.

[26] ALIDADI, A.; ARAB, S.; ASKARI, T. A novel optimized routing algorithm for qos traffic engineering in sdn-based mobile networks. *ICT Express*, v. 8, n. 1, p. 130–134, 2022. ISSN 2405-9595. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2405959521001752>.

[27] NI, H.; GUO, Z.; LI, C.; DOU, S.; YAO, C.; BAKER, T. Network coding-based resilient routing for maintaining data security and availability in software-defined networks. *Journal of Network and Computer Applications*, p. 103372, 2022. ISSN 1084-8045. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1084804522000364>.

[28] NOURA, H. N.; MELKI, R.; CHEHAB, A. Network coding and mptcp: Enhancing security and performance in an sdn environment. *Journal of Information Security and Applications*, v. 66, p. 103165, 2022. ISSN 2214-2126. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2214212622000515>.

[29] DEB, R.; ROY, S. A comprehensive survey of vulnerability and information security in sdn. *Computer Networks*, v. 206, p. 108802, 2022. ISSN 1389-1286. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1389128622000299>.

[30] AZAB, M.; SAMIR, M.; SAMIR, E. mystify: A proactive moving-target defense for a resilient sdn controller in software defined cps. *Computer Communications*, v. 189, p. 205–220, 2022. ISSN 0140-3664. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0140366422000949>.

[31] Correa Chica, J. C.; IMBACHI, J. C.; Botero Vega, J. F. Security in sdn: A comprehensive survey. *Journal of Network and Computer Applications*, v. 159, p. 102595, 2020. ISSN 1084-8045. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1084804520300692>.

[32] YUNGAICELA-NAULA, N. M.; VARGAS-ROSALES, C.; PéREZ-DíAZ, J. A.; ZAREEI, M. Towards security automation in software defined networks. *Computer Communications*, v. 183, p. 64–82, 2022. ISSN 0140-3664. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0140366421004436>.

[33] KIM, Y.; YU, J.-Y.; LEE, E.; KIM, Y.-G. Video anomaly detection using cross u-net and cascade sliding window. *Journal of King Saud University - Computer and Information Sciences*, 2022. ISSN 1319-1578. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1319157822001392>.

[34] KIM, Y.; YU, J.-Y.; LEE, E.; KIM, Y.-G. Video anomaly detection using cross u-net and cascade sliding window. *Journal of King Saud University - Computer and Information Sciences*, 2022. ISSN 1319-1578. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1319157822001392>.

[35] PEI, J.; ZHONG, K.; JAN, M. A.; LI, J. Personalized federated learning framework for network traffic anomaly detection. *Computer Networks*, v. 209, p. 108906, 2022. ISSN 1389-1286. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1389128622001001>.

[36] FERNANDES, G.; RODRIGUES, J. J.; CARVALHO, L. F.; AL-MUHTADI, J. F.; PROENcA, M. L. A comprehensive survey on network anomaly detection. *Telecommun. Syst.*, Kluwer Academic Publishers, USA, v. 70, n. 3, p. 447489, mar 2019. ISSN 1018-4864. Disponível em: <https://doi.org/10.1007/s11235-018-0475-8>.

[37] ABDALLAH, E. E.; ELEISAH, W.; OTOOM, A. F. Intrusion detection systems using supervised machine learning techniques: A survey. *Procedia Computer Science*, v. 201, p. 205–212, 2022. ISSN 1877-0509. The 13th International Conference on Ambient Systems, Networks and Technologies (ANT) / The 5th International Conference on Emerging Data and Industry 4.0 (EDI40). Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877050922004422>.

[38] YANG, Z.; LIU, X.; LI, T.; WU, D.; WANG, J.; ZHAO, Y.; HAN, H. 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://www.sciencedirect.com/science/article/pii/S0167404822000736>.

[39] SCARANTI, G. F.; CARVALHO, L. F.; BARBON, S.; PROENçA, M. L. Artificial immune systems and fuzzy logic to detect flooding attacks in software-defined networks. *IEEE Access*, v. 8, p. 100172–100184, 2020.

[40] GADGE, J.; PATIL, A. A. Port scan detection. In: *2008 16th IEEE International Conference on Networks*. [S.l.: s.n.], 2008. p. 1–6.

[41] CATILLO, M.; PECCHIA, A.; VILLANO, U. No more dos? an empirical study on defense techniques for web server denial of service mitigation. *Journal of Network*

and Computer Applications, v. 202, p. 103363, 2022. ISSN 1084-8045. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1084804522000303>.

[42] SALEEM, R.; YUAN, B.; KURUGOLLU, F.; ANJUM, A.; LIU, L. Explaining deep neural networks: A survey on the global interpretation methods. *Neurocomputing*, v. 513, p. 165–180, 2022. ISSN 0925-2312. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0925231222012218>.

[43] LYU, S.-H.; WANG, L.; ZHOU, Z.-H. Improving generalization of deep neural networks by leveraging margin distribution. *Neural Networks*, v. 151, p. 48–60, 2022. ISSN 0893-6080. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0893608022000971>.

[44] ABUQADDOM, I.; MAHAFZAH, B. A.; FARIS, H. Oriented stochastic loss descent algorithm to train very deep multi-layer neural networks without vanishing gradients. *Knowledge-Based Systems*, v. 230, p. 107391, 2021. ISSN 0950-7051. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950705121006535>.

[45] XU, Y.; ZHANG, H. Convergence of deep convolutional neural networks. *Neural Networks*, v. 153, p. 553–563, 2022. ISSN 0893-6080. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0893608022002465>.

[46] ZHAI, N.; YAO, P.; ZHOU, X. Multivariate time series forecast in industrial process based on xgboost and gru. In: *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*. [S.l.: s.n.], 2020. v. 9, p. 1397–1400.

[47] ZHAO, Y.; BAN, Y.; NASCETTI, A. Early detection of wildfires with goes-r time-series and deep gru network. In: *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*. [S.l.: s.n.], 2021. p. 3765–3768.

[48] LI, L.; HUANG, S.; OUYANG, Z.; LI, N. A deep learning framework for non-stationary time series prediction. In: *2022 3rd International Conference on Computer Vision, Image and Deep Learning International Conference on Computer Engineering and Applications (CVIDL ICCEA)*. [S.l.: s.n.], 2022. p. 339–342.

[49] SU, R.; HUANG, W.; MA, H.; SONG, X.; HU, J. Sge net: Video object detection with squeezed gru and information entropy map. In: *2021 IEEE International Conference on Image Processing (ICIP)*. [S.l.: s.n.], 2021. p. 689–693.

[50] NAIDU, T. A.; KUMAR, S. Impact of deep learning models on hate speech detection. In: *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. [S.l.: s.n.], 2021. p. 1–5.

[51] RAWAT, U.; SINGH, S. Automatic music generation: Comparing lstm and gru. In: *2022 3rd International Conference on Intelligent Engineering and Management (ICIEM)*. [S.l.: s.n.], 2022. p. 693–698.

[52] DUAN, L.; REN, Y.; DUAN, F. Adaptive stochastic resonance based convolutional neural network for image classification. *Chaos, Solitons Fractals*, v. 162, p. 112429, 2022. ISSN 0960-0779. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0960077922006397>.

[53] HUANG, X.; YE, Y.; DING, W.; YANG, X.; XIONG, L. Multi-mode dynamic residual graph convolution network for traffic flow prediction. *Information Sciences*, v. 609, p. 548–564, 2022. ISSN 0020-0255. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0020025522006971>.

[54] ZHOU, D.; WANG, B. Battery health prognosis using improved temporal convolutional network modeling. *Journal of Energy Storage*, v. 51, p. 104480, 2022. ISSN 2352-152X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2352152X22005023>.

[55] RéMY, P. *Keras-tcn*. 2019. Https://github.com/philipperemy/keras-tcn/. Acessado pela última vez: 07-11-2022.

[56] JIA, B.; LIU, J.; FENG, T.; HUANG, B.; BAKER, T.; TAWFIK, H. Ttsl: An indoor localization method based on temporal convolutional network using time-series rssi. *Computer Communications*, v. 193, p. 293–301, 2022. ISSN 0140-3664. Disponível em: <https://www.sciencedirect.com/science/article/pii/S014036642200250X>.

[57] DUN, A.; YANG, Y.; LEI, F. Dynamic graph convolution neural network based on spatial-temporal correlation for air quality prediction. *Ecological Informatics*, v. 70, p. 101736, 2022. ISSN 1574-9541. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1574954122001868>.

[58] TATLICIOGLU, E.; YILMAZ, B. M.; SAVRAN, A.; ALCI, M. Adaptive fuzzy logic with self-adjusting membership functions based tracking control of surface vessels. *Ocean Engineering*, v. 253, p. 111129, 2022. ISSN 0029-8018. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0029801822005406>.

[59] DELLANNA, D.; JAMSHIDNEJAD, A. Evolving fuzzy logic systems for creative personalized socially assistive robots. *Engineering Applications of Artificial Intelligence*, v. 114, p. 105064, 2022. ISSN 0952-1976. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0952197622002251>.

[60] MIHOUB, A.; FREDJ, O. B.; CHEIKHROUHOU, O.; DERHAB, A.; KRICHEN, M. Denial of service attack detection and mitigation for internet of things using looking-back-enabled machine learning techniques. *Computers Electrical Engineering*, v. 98, p. 107716, 2022. ISSN 0045-7906. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0045790622000337>.

[61] ZENG, Z.; PENG, W.; ZENG, D.; ZENG, C.; CHEN, Y. Intrusion detection framework based on causal reasoning for ddos. *Journal of Information Security and Applications*, v. 65, p. 103124, 2022. ISSN 2214-2126. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2214212622000163>.

[62] GAURAV, A.; GUPTA, B. B.; PANIGRAHI, P. K. A novel approach for ddos attacks detection in covid-19 scenario for small entrepreneurs. *Technological Forecasting and Social Change*, v. 177, p. 121554, 2022. ISSN 0040-1625. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0040162522000865>.

[63] MANASRAH, A. M.; KHDOUR, T.; FREEHAT, R. Dga-based botnets detection using dns traffic mining. *Journal of King Saud University - Computer and Information Sciences*, v. 34, n. 5, p. 2045–2061, 2022. ISSN 1319-1578. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1319157822000726>.

[64] ASSIS, M. V.; CARVALHO, L. F.; LLORET, J.; PROENçA, M. L. 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://www.sciencedirect.com/science/article/pii/S1084804520304008>.

[65] MARTINS, I.; RESENDE, J. S.; SOUSA, P. R.; SILVA, S.; ANTUNES, L.; GAMA, J. Host-based ids: A review and open issues of an anomaly detection system in iot. *Future Generation Computer Systems*, v. 133, p. 95–113, 2022. ISSN 0167-739X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167739X22000760>.

[66] CHATTERJEE, A.; AHMED, B. S. Iot anomaly detection methods and applications: A survey. *Internet of Things*, v. 19, p. 100568, 2022. ISSN 2542-6605. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2542660522000622>.

[67] SGUEGLIA, A.; Di Sorbo, A.; VISAGGIO, C. A.; CANFORA, G. A systematic literature review of iot time series anomaly detection solutions. *Future Generation Computer Systems*, v. 134, p. 170–186, 2022. ISSN 0167-739X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167739X22001285>.

[68] DING, Q.; LI, J. Anogla: An efficient scheme to improve network anomaly detection. *Journal of Information Security and Applications*, v. 66, p. 103149, 2022. ISSN 2214-2126. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2214212622000394>.

[69] WANG, C.; ZHOU, H.; HAO, Z.; HU, S.; LI, J.; ZHANG, X.; JIANG, B.; CHEN, X. Network traffic analysis over clustering-based collective anomaly detection. *Computer Networks*, v. 205, p. 108760, 2022. ISSN 1389-1286. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1389128622000019>.

[70] DERHAB, A.; ALDWEESH, A.; EMAM, A. Z.; KHAN, F. A. Intrusion detection system for internet of things based on temporal convolution neural network and efficient feature engineering. *Wireless Communications and Mobile Computing*, Hindawi, v. 2020, p. 6689134, Dec 2020. ISSN 1530-8669. Disponível em: <https://doi.org/10.1155/2020/6689134>.

[71] NOVAES, M. P.; CARVALHO, L. F.; LLORET, J.; PROENçA, M. L. 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://www.sciencedirect.com/science/article/pii/S0167739X21002429>.

[72] AUDIBERT, J.; MICHIARDI, P.; GUYARD, F.; MARTI, S.; ZULUAGA, M. A. Do deep neural networks contribute to multivariate time series anomaly detection? *Pattern Recognition*, v. 132, p. 108945, 2022. ISSN 0031-3203. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0031320322004253>.

[73] LINDEMANN, B.; MASCHLER, B.; SAHLAB, N.; WEYRICH, M. A survey on anomaly detection for technical systems using lstm networks. *Computers in Industry*, v. 131, p. 103498, 2021. ISSN 0166-3615. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0166361521001056>.

[74] AMADI, P.; IKOT, A.; OKORIE, U.; OBAGBOYE, L.; RAMPHO, G.; HORCHANI, R.; ONYEAJU, M.; ALREBDI, H.; ABDEL-ATY, A.-H. Shannon entropy and complexity measures for bohr hamiltonian with triaxial nuclei. *Results in Physics*, p. 105744, 2022. ISSN 2211-3797. Disponível em: <https://www.sciencedirect.com/science/article/pii/S221137972200417X>.

[75] PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

[76] CHOLLET, F. et al. *Keras*. 2015. Https://keras.io. Acessado pela última vez: 04-05-2022.

[77] AMIDAN, B.; FERRYMAN, T.; COOLEY, S. Data outlier detection using the chebyshev theorem. In: *2005 IEEE Aerospace Conference*. [S.l.: s.n.], 2005. p. 3814–3819.

[78] DATASETS used in publications - Orion Research Group. Http://www.uel.br/grupos/orion/datasets.html. Acessado pela última vez: 04-05-2022.

[79] MININET: An Instant Virtual Network on your Laptop (or other PC). Http://mininet.org/. Acessado pela última vez: 04-05-2022.

[80] SCAPY - Packet crafting for Python2 and Python3. Https://scapy.net. Acessado pela última vez: 04-05-2022.

[81] HPING3 - command-line oriented TCP/IP packet assembler/analyzer. Http://hping.org. Acessado pela última vez: 04-05-2022.

[82] de Assis, M. V.; CARVALHO, L. F.; RODRIGUES, J. J.; LLORET, J.; Proença Jr, M. L. 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://www.sciencedirect.com/science/article/pii/S0045790620305930>.

[83] SCARANTI, G. F.; CARVALHO, L. F.; BARBON, S.; PROENçA, M. L. Artificial immune systems and fuzzy logic to detect flooding attacks in software-defined networks. *IEEE Access*, v. 8, p. 100172–100184, 2020.

[84] SHARAFALDIN, I.; LASHKARI, A. H.; HAKAK, S.; GHORBANI, A. A. Developing realistic distributed denial of service (ddos) attack dataset and taxonomy. In: *2019 International Carnahan Conference on Security Technology (ICCST)*. [S.l.: s.n.], 2019. p. 1–8.

[85] MACIá-FERNáNDEZ, G.; CAMACHO, J.; MAGáN-CARRIóN, R.; GARCíA-TEODORO, P.; THERóN, R. Ugr16: A new dataset for the evaluation of cyclostationarity-based network idss. *Computers Security*, v. 73, p. 411–424, 2018. ISSN 0167-4048. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167404817302353>.

TRABALHOS PUBLICADOS PELO AUTOR

1. D. M. Brandão Lent, M. P. Novaes, L. F. Carvalho, J. Lloret, J. J. P. C. Rodrigues and M. L. Proença, "**A Gated Recurrent Unit Deep Learning Model to Detect and Mitigate Distributed Denial of Service and Portscan Attacks**," in IEEE Access, vol. 10, pp. 73229-73242, 2022, doi: 10.1109/ACCESS.2022.3190008, IF: 3,367.