Universidade Estadual de Londrina
Centro de Tecnologia e Urbanismo
Departamento de Engenharia Elétrica

Matheus Pereira de Novaes

Sistema de Detecção de Intrusão baseado em Anomalias de Redes utilizando Redes Neurais Profundas

Londrina
2023

Universidade Estadual de Londrina

Centro de Tecnologia e Urbanismo
Departamento de Engenharia Elétrica

Matheus Pereira de Novaes

Sistema de Detecção de Intrusão baseado em Anomalias de Redes utilizando Redes Neurais Profundas

Tese orientada pelo Prof. Dr. Mario Lemes Proença Jr. intitulada "Sistema de Detecção de Intrusão baseado em Anomalias de Redes utilizando Redes Neurais Profundas" e apresentada ao Programa de Pós-Graduação associado em Engenharia Elétrica da Universidade Estadual de Londrina, como requisito parcial para a obtenção do Título de Doutor em Engenharia Elétrica.

Orientador: Prof. Dr. Mario Lemes Proença Jr.
Coorientador: Prof. Dr. Jaime Lloret Mauri

Londrina
2023

Ficha Catalográfica

Matheus Pereira de Novaes
Sistema de Detecção de Intrusão baseado em Anomalias de Redes utilizando Redes Neurais Profundas - Londrina, 2023 - 157 p., 30 cm.
Orientador: Prof. Dr. Mario Lemes Proença Jr.
1. Aprendizado Profundo. 2. Detecção de Anomalia. 3. Redes Definidas por Software 4. LSTM. 5. Lógica Fuzzy. 6. GAN.
I. Universidade Estadual de Londrina. Curso de Doutorado em Engenharia Elétrica. II. Sistema de Detecção de Intrusão baseado em Anomalias de Redes utilizando Redes Neurais Profundas.

Matheus Pereira de Novaes

Sistema de Detecção de Intrusão baseado em Anomalias de Redes utilizando Redes Neurais Profundas

Tese apresentada ao Programa de Pós-Graduação em Doutorado em Engenharia Elétrica da Universidade Estadual de Londrina, como requisito para a obtenção do título de Doutor em Engenharia Elétrica

Comissão Examinadora

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

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

Prof. Dr. José Palazzo Moreira de Oliveira
Universidade Federal do Rio Grande do Sul

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

Prof. Dr. Leonimer Flávio de Melo
Universidade Estadual de Londrina

Londrina, 19 de abril de 2023

Dedico este trabalho a todos aqueles que, de alguma forma,
auxiliaram para na concretização desta etapa.

Agradecimentos

A Deus, por ter me dado discernimento e sabedoria ao longo desta jornada. Sem ele, acredito que não teria alcançando meus objetivos.

À minha família, por ter me dado todo o suporte necessário e sempre estar ao meu lado. Agradeço à minha namorada, que teve muita compreensão e ofereceu grande apoio durante esta fase.

Ao meu orientador Prof. Dr. Mario Lemes Proença Jr, pela oportunidade e confiança, me aceitando como orientando desde a iniciação científica até o Doutorado. Obrigado por ter compartilhado seu tempo e seus conhecimentos e mostrar que o trabalho duro gera bons frutos.

Aos membros do grupo de pesquisa de redes pelo apoio e pela contribuição para a condução deste trabalho.

A todos os professores e técnicos da UEL que, de alguma forma, contribuíram para a minha formação acadêmica.

A FA/CAPES, pelos 25 meses de bolsa concedida.

"Mas buscai primeiro o Reino de Deus, e a sua justiça, e todas essas coisas vos serão acrescentadas."

(Bíblia Sagrada, Mateus 6, 33)

Matheus Pereira de Novaes. **Sistema de Detecção de Intrusão baseado em Anomalias de Redes utilizando Redes Neurais Profundas**. 2023. 157 p. Tese de Doutorado em Engenharia Elétrica - Universidade Estadual de Londrina, Londrina-PR.

Resumo

Nos últimos anos, com a introdução de novos dispositivos, tais como os introduzidos pelo paradigma de Internet das Coisas (*Internet of Things - IoT*), os sistemas de redes de computadores têm se tornado estruturas complexas de gerenciamento e controle. A principal razão para isso é a quantidade de dispositivos heterogêneos que compõem a rede. O paradigma de Redes Definidas por Software (*Software-Defined Networking - SDN*) introduziu ferramentas para simplificar a configuração e o gerenciamento, além de possibilitar inovações mais significativas em redes de comunicação. A *SDN* permite o gerenciamento centralizado da rede, cujo controle é dissociado do plano de encaminhamento e centralizado em um controlador. A centralização da lógica de controle pode se tornar um alvo ideal para ataques de agentes maliciosos, principalmente os ataques distribuídos de negação de serviço (*Distributed Denial of Service - DDoS*). Consequentemente, é indispensável o emprego de mecanismos de defesa para garantir a operabilidade da rede, aplicando técnicas para a detecção e a mitigação de ataques. Esta tese tem como objetivo explorar a capacidade de generalização dos métodos de *Deep Learning* para propor e desenvolver uma solução arquitetural de detecção e mitigação de anomalias em redes *SDN*. Para isso, dois sistemas modulares de detecção de anomalias foram desenvolvidos. No primeiro, foi empregada uma arquitetura de rede neural profunda recorrente, a *Long Short-Term Memory* (*LSTM*), em conjunto com a lógica Fuzzy para detecção de ataques de *Portscan* e *DDoS*. O segundo sistema foi aplicado treinamento *Adversarial*, que usa o *framework Generative Adversarial Network* (*GAN*) para detectar ataques *DDoS* recentes. O desempenho dos sistemas propostos foi avaliado em diferentes cenários e comparado com outros métodos, presentes na literatura e desenvolvidos com o mesmo propósito. Por meio do emprego de métricas e testes de desempenho, os resultados obtidos demonstram que os sistemas foram eficientes na detecção e na mitigação da ocorrência de eventos anômalos nos cenários avaliados.

Palavras-Chave: 1. Aprendizado Profundo. 2. Detecção de Anomalia. 3. Redes Definidas por Software 4. LSTM. 5. Lógica Fuzzy. 6. GAN.

Matheus Pereira de Novaes. **Intrusion Detection System based on Network Anomalies using Deep Neural Networks**. 2023. 157 p. Thesis in Electrical Engineering - Londrina State University, Londrina-PR.

Abstract

Over the last few years, with the introduction of new devices, such as the Internet of Things - IoT, computer network systems have become complex structures for management and control. The leading reason for this is due to the number of heterogeneous devices that make up the network. The Software-Defined Networking paradigm (SDN) introduced tools to simplify configuration and management, in addition to enabling more significant innovation in communication networks. The SDN enables centralized network management, in which the network control is dissociated from the data forwarding plane and centralized in a controller. The centralization of the control plane provides a global view of the network topology. However, centralizing control logic can be an ideal target for malicious agents attacks, mainly Distributed Denial of Service (DDoS) attacks. Consequently, it is essential to use defense mechanisms to guarantee the network's operability, applying techniques for detecting and mitigating attacks. Therefore, this thesis aims to explore the generalization capacity of Deep Learning methods to propose and develop an anomaly detection and mitigation solution in SDN networks. For this, two modular anomaly detection systems were developed. In the first system, a recurrent deep neural architecture was used, Long Short-Term Memory (LSTM) together with Fuzzy logic to detect Portscan and DDoS attacks. The second system applied Adversarial training, which uses the Generative Adversarial Network framework GAN to detect recent DDoS attacks. The performance of the proposed systems was evaluated in different scenarios and compared with other methods present in the literature that were developed with the same purpose. Through the use of metrics and performance tests, the results obtained demonstrate that the systems were efficient in detecting and mitigating the occurrence of anomalous events in the evaluated scenarios.

Key-words: 1. Deep Learning. 2. Anomaly Detection. 3. Software-Defined Network. 4. LSTM. 5. Fuzzy Logic. 6. GAN.

Lista de ilustrações

Lista de tabelas

Lista de siglas e abreviaturas

ACSC	*Australian Cyber Security Center*
AE	*Auto-Encoder*
AUC	*AreaUnder the Curve*
BM	*Boltzmann Machines*
CC	*Complexidade Computacional*
CNN	*Convolutional Neural Network*
DAE	*Denoising Auto-Encoder*
DBM	*Deep Boltzmann machine*
DDoS	*Distributed Denial of Service*
DL	*Deep Learning*
DNN	*Deep Neural Network*
DNS	*Domain Name System*
DoS	*Denial of Service*
DSNS	*Digital Signature of Network Segments*
DSNSF	*Digital Signature of Network Segment using Flow analysis*
DT	*Decision Tree*
ECA	Evento-Condição-Ação
FN	*False Negative*
FP	*False Positive*
FPR	*False Positive Rate*
GA	*Genetic Algorithm*
GANs	*Generative Adversarial Networks*
GB	*Gradient Boosting*
GRU	*Gated Recurrent Unit*
HIDS	*Host-based Intrusion Detection System*
IDS	*Intrusion Detection System*
IoT	*Internet of Things*
IP	*Internet Protocol*
kNN	*k-Nearest Neighbor*
LDAP	*Lightweight Directory Access Protocol*
LR	*Logistic Regression*
LSTM	*Long Short-Term Memory*
MAE	*Mean Absolute Error*
MAPE	*Mean Absolute Percentage Error*
ML	*Machine Learning*

MLP	*Multilayer Perceptron*
MSE	*Mean Squared Error*
MSSQL	*Microsoft Structured Query Language*
NDAE	*Non-symmetric Deep Auto-Encoder*
NIDS	*Network Intrusion Detection System*
NTP	*Network Time Protocol*
PSO-DS	*Particle Swarm Optimization for Digital Signature*
RBM	*Restricted Boltzmann Machine*
RF	*Random Forest*
RMSE	*Root Mean Squared Error*
RNA	*Redes Neurais Artificias*
RNN	*Recurrent Neural Network*
ROC	*Receiver Operating Characteristics*
RWO	*Remora Whale Optimization*
SAE	*Stacked Auto-Encoder*
SDN	*Software-Defined Networking*
SNMP	*Simple Network Management Protocol*
SSDP	*Simple Service Discovery Protocol*
SVC	*Support Vector Classifier*
SVM	*Support Vector Machine*
SYN	*Synchronize*
TFTP	*Trivial File Transfer Protocol*
TN	*True Negative*
TP	*True Positive*
UDP	*User Datagram Protocol*

Lista de símbolos

β_t	Conjunto de registros fluxos coletados
α_i	Registro de fluxo
$x^{bits}_{\in\alpha_i}$	Quantidade de *bits* pertencentes ao fluxo α_i
$x^{pacotes}_{\in\alpha_i}$	Quantidade de pacotes pertencentes ao fluxo α_i
$x^{srcIP}_{\in\alpha_i}$	Endereço de IP de origem
$x^{srcPort}_{\in\alpha_i}$	Número da porta de origem
$x^{dstIP}_{\in\alpha_i}$	Endereço de IP de destino
$x^{dstPort}_{\in\alpha_i}$	Número da porta de destino
t	Instante de tempo
$\mathbf{f}_t$	*forget gate*
$\mathbf{i}_t$	*input gate*
$\mathbf{o}_t$	*output gate*
$\mathbf{c'}_t$	*candidate value*
$\mathbf{W}$	Matrizes de pesos sinápticos *input gate*
$\mathbf{U}$	Matrizes de pesos sinápticos camadas recorrentes
$\mathbf{u}$	Número de unidades ocultas
$\mathbf{n}$	Tamanho do vetor de entrada
$\mathbf{b}$	Vetor de *bias*
$\mathbf{h}_{t-1}$	Saída anterior da LSTM
$\sigma(\cdot)$	Função de ativação Relu
$\tanh(\cdot)$	Função de ativação Tangente Hiperbólica
$\odot$	Produto Hadamard
$H(f)$	Entropia de Shannon
$\mathbf{X}$	Conjunto de dados de fluxos
y_i	Previsão do i-ésimo atributo de fluxo
μ	Média
k	Parâmetro de desvios padrões
ϕ	Desvio padrão
f	Função pertinência
$\mathbf{x}_t$	Tráfego real
y_t	Tráfego previsto
$\hat{\sigma}_t$	Limiar de normalidade
f_j	Grau de pertinência para o atributo de fluxo j
f'_j	Score de anomalia do atributo de fluxo j
γ	Limiar de anomalia para a detecção de ataques de *Portscan*

ζ	Limiar de anomalia para a detecção de ataques de *DDoS*
$\mathcal{O}(\cdot)$	Complexidade Computacional
G	Modelo generativo
D	Modelo discriminativo
p_z	Ruído aleatório
p_g	Distribuição generativa
p_{data}	Distribuição dos dados reais
d_{epoch}	Número de épocas para alternar e atualizar o gerador e o discriminador
m	Tamanho do *mini-batch*

Sumário

1 Introdução

Ao longo dos anos, o avanço do protocolo *Ethernet* vem possibilitando que as informações trafeguem pela rede em altas taxas de transmissão, atingindo 400 *Gigabit/s*. Com esse avanço e a introdução da Internet das Coisas (*Internet of Things - IoT*), a quantidade de aplicações e serviços que utilizam a internet tem crescido rapidamente. As redes de computadores têm se tornado cada vez mais complexas devido aos vários elementos que as compõem como, por exemplo, *firewall*, Sistema de Dectecção de Intrusão (*Intrusion Detection System - IDS*), balanceador de carga, *switches*, roteadores, etc. Na arquitetura tradicional, cada ativo de rede utiliza protocolos complexos e a sua configuração difere entre os fabricantes. A arquitetura tradicional de gerenciamento de rede torna-se inadequada em razão da heterogeneidade desses ativos, principalmente nos *datacenters* atuais, que oferecem serviços de *cloud computing* e empregam tecnologias de virtualização [1, 2, 3, 4].

Apesar de as Redes Definidas por Software (*Software-Defined Networking - SDN*) não terem sido criadas com um objetivo específico para as funções de virtualização, elas são uma arquitetura emergente para projetar redes futuras e satisfazer as novas demandas das aplicações existentes [4]. Esse paradigma introduziu ferramentas para simplificar a configuração e o gerenciamento, além de permitir uma inovação mais significativa nas redes de comunicação. A principal característica da arquitetura *SDN* é a separação entre o plano de controle e o plano de dados, isto é, o plano de controle é desacoplado do dispositivo de rede e centralizado em um controlador [5]. A centralização do plano de controle fornece uma visão global da topologia de rede. Além disso, permite a manipulação do fluxo do tráfego em tempo de execução, por meio de uma interface de *software* aberta e bem definida para o gerenciamento do plano de dados.

As redes *SDN* introduziram recursos de programação e centralização da lógica de controle que facilitam seu gerenciamento e sua configuração. Porém, esses recursos não eliminaram as vulnerabilidades relacionadas à segurança [6]. A quantidade de ataques aumentou em número e na sofisticação em que são executados pelos agentes maliciosos, principalmente o ataque de Negação de Serviço Distribuído (*Distributed Denial of Service - DDoS*) [7, 8]. O ataque de *DDoS* tem como finalidade o esgotamento de algum recurso, seja com relação ao servidor - em que o atacante, mediante inúmeras solicitações, busca indisponibilizar algum tipo de serviço - com relação à infraestrutura, saturando um *link* de rede [9]. De acordo com os relatórios de ameaças do Australian Cyber Security Center (ACSC) e da empresa McAfee, os ataques de *DDoS* são os que ocorrem com maior frequência em relação aos demais ataques reportados [2]. No ano de 2020, a Amazon reportou o maior ataque de *DDoS* já registrado, com de 2.3 *Tbps* e duração de três dias [10].

O gerenciamento dos fluxos na rede *SDN* é executado de forma centralizada, que além de beneficiar de tal tal forma o gerenciamento, cria um ponto crítico de vulnerabilidade, podendo ser alvo de ataques de *DDoS*, *Portscan*, falsificação de *IP* (*Internet Protocol*) etc [11, 12, 13]. Portanto, a segurança das redes *SDN* permanece vulnerável e se faz necessário o desenvolvimento de soluções relacionadas à detecção e à mitigação de ataques.

Mecanismos de segurança são aplicados para detectar e impedir as ações de agentes maliciosos. Para esse propósito, sistemas de detecção de intrusão de rede (*Network Intrusion Detection System - NIDS*) são amplamente aplicados. Os *NIDS* fornecem um conjunto de ferramentas capazes de reconhecer automaticamente comportamentos anormais de rede. Nos *NIDS* baseados em anomalias, o objetivo é gerar um perfil do comportamento normal da rede com base em dados históricos. A principal vantagem dessa abordagem é a detecção de ataques desconhecidos. Com o aumento das ameaças de segurança e a alta frequência com que novos padrões de ataques surgem, a abordagem baseada em anomalias se torna eficiente para a detecção desses tipos de ataque.

A detecção de anomalias não é uma atividade trivial, principalmente com o aumento das ameaças de segurança e com o enorme volume de tráfego [2, 14]. Para esse propósito, é necessário o emprego de técnicas eficientes para melhorar o desempenho dos sistemas de detecção de anomalias. Por conta de seu poder preditivo e de sua capacidade de generalização, os métodos de Aprendizagem de Máquina (*Machine Learning - ML*) têm sido explorados em diversos trabalhos [15, 16, 17]. Porém, os métodos clássicos de *ML* apresentam alguns problemas, tendo como exemplo o sobreajuste (*overfitting*), devido à presença de atributos irrelevantes ou redundantes, e a distribuição desbalanceada do tráfego de rede [18]. O poder preditivo dos métodos depende em grande parte dos atributos utilizados no processo de treinamento, sendo necessária uma análise extensa para capturar os atributos e as estatísticas mais relevantes do tráfego [19]. Recentemente, com a exploração do conceito de Aprendizagem Profunda (*Deep Learning - DL*), diversos modelos de *DL* foram empregados para a detecção de anomalias, devido à capacidade de aprendizagem e à generalização das soluções sobre os atributos empregados [18, 20].

Apresenta-se como um dos objetivos principais desta pesquisa o desenvolvimento de um sistema modular para a detecção e a mitigação de anomalias em ambientes de redes *SDN* utilizando redes neurais profundas. O primeiro módulo é responsável pela coleta dos registros de fluxos dos *switches* utilizando o protocolo OpenFlow. O segundo módulo aborda o processamento que é responsável por extrair características específicas que auxiliam na caracterização do tráfego de rede. O terceiro módulo atua na caracterização e a detecção de anomalias, empregando a arquitetura de rede neural profunda, a *Long Short-Term Memory* (*LSTM*), para prever o comportamento normal do tráfego de rede e a lógica Fuzzy no processo de detecção. O quarto módulo do sistema é responsável pela mitigação das anomalias detectadas, pretendendo minimizar os danos causados por um atacante.

Os métodos de *DL*, tais como as redes convolucionais [21] e as redes recorrentes [22], podem facilmente extrair e aprender características e padrões do comportamento da rede, fornecendo informações úteis para a detecção de ataques. Além disso, com o aumento rápido na quantidade de usuários, o tráfego de rede torna-se complexo e as redes neurais profundas são poderosas na redução da análise da complexidade do tráfego de rede devido à sua capacidade de aprender representações de dados massivamente complexas e de lidar com isso sem esforços humanos [23, 24].

Apesar das inúmeras aplicações que utilizam modelos de *DL* em *IDS*, estudos demonstram que redes neurais profundas são sensíveis a exemplos adversários (*adversarial examples*) executados por agentes maliciosos para fazer com que os métodos de *DL* cometam erros na detecção de ataques [25, 26, 27]. Exemplos adversários são amostras com ruídos projetadas intencionalmente por um invasor para que uma rede neural profunda as classifique incorretamente, ou seja, fazer um ataque ser classificado como uma amostra normal. Esses ataques podem diminuir significativamente a robustez dos modelos de aprendizagem profunda e aumentar os problemas de segurança do modelo [28]. Portanto, é extremamente importante fornecer robustez aos métodos de *DL* aplicados em sistemas de detecção de anomalias.

De acordo com X. Zhang *et al.* [29], o treinamento adversário (*Adversarial Training*) pode ser aplicado para proteger o sistema contra ameaças de exemplos adversários. Uma das principais técnicas presentes na literatura utilizada para treinamento adversário são as Redes Adversárias Generativas (*Generative Adversarial Networks - GANs*) [30, 31]. Propostas por Goodfellow *et al.* [32], as redes *GAN* permitem gerar exemplos de adversários ao treinar um gerador e um discriminador simultaneamente de forma concorrente. Assim, o discriminador melhora as taxas de detecção usando os exemplos adversários gerados e se atualiza contra eles. Em termos práticos, isso significa adicionar exemplos de dados adversários ao conjunto de dados original e usá-lo na fase de treinamento do modelo.

Considerando a importância e a contínua necessidade de melhorias e de aprimorar a robustez e a eficácia dos *NIDS* que empregam métodos de *DL* como núcleo principal na detecção de anomalias, é indispensável o emprego de ferramentas para dar suporte às atividades de gerenciamento e segurança. Além disso, essas ferramentas devem operar de modo automatizado para facilitar a identificação da ocorrência de eventos anômalos e tomar contramedidas a fim de minimizar os efeitos causados por agentes maliciosos. Dessa forma, é apresentado um sistema de detecção de anomalias baseado em treinamento adversário aplicando a *Generative Adversarial Network* para a detecção e a defesa contra ataques *DDoS* atualizados.

Este trabalho tem como objetivo explorar a capacidade de generalização em aprender e extrair padrões de curto e longo prazo do método *LSTM* e o potencial do *framework GAN* contra ataques adversários para propor e desenvolver uma solução de detecção e mitigação de anomalias em redes *SDN*, destacando-se as seguintes contribuições:

- Proposta de uma solução de sistema de defesa modular aplicado em ambientes *SDN*, incluindo os módulos de coleta e processamentos de fluxos *IP*, caraterização do tráfego de rede e detecção de eventos anômalos e mitigação.
- Emprego do modelo de rede neural profunda *Long Short-Term Memory* (*LSTM*) para a caracterização do tráfego de rede.
- Proposta e implementação de um mecanismo de defesa para a detecção de ataques de *DDoS* e *Portscan* utilizando o sistema de Inferência Fuzzy.
- Detecção e defesa contra ataques *DDoS* adversários por meio de uma abordagem *Adversarial Deep Learning*, que fornece uma taxa de detecção mais precisa e menos sensível a exemplos adversários.
- Coleta e análise do tráfego de rede a cada segundo, permitindo que o sistema de detecção de anomalias atue em tempo quase real (*near real-time*): quanto menor esse tempo, menor será o tempo de resposta do sistema na detecção das ameaças recebidas.
- Detecção de diferentes tipos de ataque de *DDoS*, por exemplo, *Network Time Protocol* (*NTP*), *Domain Name System* (*DNS*), *Lightweight Directory Access Protocol* (*LDAP*), *Microsoft Structured Query Language* (*MSSQL*), *Network Basic Input/Output System* (*NetBIOS*), *Simple Network Management Protocol* (*SNMP*), *Simple Service Discovery Protocol* (*SSDP*), *User Datagram Protocol* (*UDP*), *UDP-Lag*, *WebDDoS* (*ARME*), *Synchronize* (*SYN*) e *Trivial File Transfer Protocol* (*TFTP*).
- Comparação da eficiência da solução proposta com diferentes métodos de aprendizagem profunda presentes na literatura para a detecção de *DDoS* em *SDN*.

O restante deste trabalho está organizado da seguinte forma: no Capítulo 2, são apresentados e analisados os trabalhos relacionados que atuam na detecção e na mitigação de anomalias em redes *SDN*. No Capítulo 3, encontram-se os fundamentos teóricos acerca dos temas centrais abordados neste trabalho. No Capítulo 4, é apresentado o sistema arquitetural modular desenvolvido, incluindo os detalhes de cada módulo e os métodos desenvolvidos. No Capítulo 5, são apresentados e detalhados os cenários de testes para a avaliação do sistema proposto; além disso, os resultados obtidos são discutidos e comparados a outros métodos presentes na literatura. Por fim, no Capítulo 6, são apresentadas as conclusões desta tese e as oportunidades de trabalhos futuros.

2 Trabalhos relacionados

Atualmente, as redes *SDN* são amplamente utilizadas, contudo apresentam vários problemas relacionados à segurança [33, 34, 35]. Portanto, a segurança das redes *SDN* permanece indefinida e soluções relacionadas à detecção e à mitigação de ataques têm sido desenvolvidas [7, 8]. A Tabela 1, no final do capítulo, apresenta um resumo dos trabalhos analisados neste capítulo, incluindo o alvo do sistema proposto, os tipos de algoritmo/técnica implementados, o plano de rede e os *datasets* utilizados.

2.1 Detecção de anomalias em ambientes tradicionais usando *Shallow Learning* e *Deep Learning*

Hamamoto *et al.* [36] propuseram um sistema de detecção de anomalias aplicado às redes de larga escala. Os autores utilizaram a abordagem de "Assinatura Digital de Segmento de Rede usando análise de fluxos" (*DSNSF*, do inglês *Digital Signature of Network Segment using Flow analysis*) para gerar assinaturas do comportamento de rede normal aplicando Algoritmo Genético (*GA*, do inglês *Genetic Algorithm*). Além disso, a lógica Fuzzy foi usada junto aos *DSNSFs* gerados para detectar comportamentos anômalos na rede analisada. Para validar o sistema proposto, foram utilizados dados reais coletados da Universidade Estadual de Londrina (UEL) por intermédio do protocolo de gerência sFlow. Com o uso de ferramentas de simulação de eventos anômalos, três anomalias diferentes foram injetadas nos dados reais da rede: *DoS*, *DDoS* e *Flash Crowd*. O sistema proposto se mostrou eficiente, com taxas de precisão acima de 96%. Diferentes trabalhos também aplicaram a abordagem de *DSNSF* por meio do uso de outras técnicas, como Holt-Winters [37] e Wavelet [38]. No entanto, as caracterizações do tráfego nesses trabalhos utilizaram uma abordagem que analisa de duas a quatro semanas de dados para reconhecimento de padrões e geração do perfil normal no ambiente tradicional de rede. Além disso, esses trabalhos apresentaram uma limitação: as detecções dos ataques foram realizadas em períodos entre um e cinco minutos. Diferentemente desses trabalhos, o modelo proposto nesta tese realiza a predição do comportamento normal do tráfego aplicando uma janela deslizante e a detecção de eventos anômalos a cada um segundo.

Jmila *et al.* [39] avaliaram a robustez de sete métodos tradicionais de *Machine Learning* para a detecção de anomalias em ambientes tradicionais: *Adaboost*, *Bagging*, *Gradient Boosting* (*GB*), *Logistic Regression* (*LR*), *Decision Tree* (*DT*), *Random Forest* (*RF*) e *Support Vector Classifier* (*SVC*). Além disso, os autores aplicaram uma técnica de defesa de aumento de dados gaussiana e avaliaram sua contribuição para melhorar a robustez dos métodos para a detecção dos ataques. Na fase de avaliação foram conduzidos experimentos

em diferentes cenários usando os conjuntos de dados NSL-KDD e UNSW-NB 15. Os resultados apresentados mostram que os ataques não têm o mesmo impacto em todos os métodos, que a robustez de um método depende do ataque sofrido e que o *trade-off* entre desempenho e robustez deve ser considerado dependendo do cenário de detecção de intrusão na rede.

Trabalhos recentes têm explorado o poder das diferentes arquiteturas de redes neurais profundas para a detecção de anomalias em ambientes tradicionais de redes de computadores. Além disso, estudos apresentam a combinação dessas arquiteturas desenvolvendo modelos híbridos. Kasim *et al.* [40] apresentaram um modelo de aprendizagem profunda híbrido que consiste em uma arquitetura composta de uma rede neural convolucional (*Convolutional Neural Network - CNN*) e uma *LSTM*. O modelo proposto fornece uma solução para a detecção de *Domain Name System* (*DNS*) *flood*, com a normalização e a seleção de *features* do conjunto de dados CICIDS. A arquitetura de aprendizagem profunda, incluindo *CNN* e *LSTM*, contribuiu para a detecção de ataques de *DNS flood* com uma alta taxa de acurácia e uma baixa taxa de falsos-positivos em comparação aos métodos de aprendizagem de máquina tradicionais.

Pingale *et al.* [41] desenvolveram um modelo profundo híbrido para a detecção de intrusão, denominado *Remora Whale Optimization* (*RWO*). Inicialmente, foi empregada uma arquitetura de *CNN* para a extração e a seleção das *features*. Por fim, os autores utilizaram um modelo híbrido profundo, usando *Deep Maxout Network* e *Deep Auto Encoder*. O processo de treinamento do modelo é feito pelo algoritmo *RWO*. O modelo proposto foi avaliado nos conjuntos de dados NSL-KDD, CICIDS2018 e UNSW-NB15 e as métricas utilizadas foram *precision*, *recall* e *f1-score*. Os resultados obtidos para as métricas avaliadas foram superiores a 90%.

Zhu *et al.* [42] projetaram e implementaram um método *black box* para a detecção de intrusão em redes tradicionais. O método desenvolvido pelos autores inclui duas etapas: primeiro, foi aplicado o *framework Generative Adversarial Network* (GAN) para gerar exemplos adversários de pacotes que podem evitar a detecção de anomalias. Em seguida, os pacotes gerados são inseridos no tráfego malicioso original para que os detectores de anomalias não identifiquem o tráfego malicioso. Na fase de detecção, foi aplicada a arquitetura de rede recorrente *Gated Recurrent Unit* (GRU). A validação do método foi realizada utilizando os conjuntos de dados Kitsune, CICIDS2017, MAWILA e UNSW-NB15. O desempenho do método proposto foi verificado ao calcular as métricas de *precision*, *recall* e *f1-score*. Os resultados experimentais demonstraram que o método tem uma taxa de detecção de 90%. Apesar de a avaliação do método proposto ter sido feita em diferentes conjuntos de dados, os autores não avaliaram ou propuseram a mitigação dos ataques, uma etapa importante na detecção de anomalias.

2.2 Detecção de anomalias em ambientes SDN usando *Shallow Learning* e *Deep Learning*

Na arquitetura *SDN*, a lógica de encaminhamento é centralizada e alocada no controlador. A centralização da lógica de encaminhamento apresenta vulnerabilidades que podem ser exploradas por agentes maliciosos, as quais estão presentes nos *links* de comunicação entre o controlador e os dispositivos de encaminhamentos e na memória dos *switches*. De acordo com Aleroud e Alsmadi [43], essas vulnerabilidades são intensificadas durante a ocorrência de ataques de negação de serviço, pois há um aumento do volume de tráfego devido às inúmeras solicitações de conexões realizadas pelos atacantes. Nos dispositivos de encaminhamento, quando estão sob ataques, a quantidade de fluxos recebidos aumenta consideravelmente. Os *switches* são dispositivos que têm memória limitada e os fluxos maliciosos recebidos podem ocupar toda a capacidade de armazenamento da tabela de encaminhamento presente nesses dispositivos. A ocupação total dessa tabela faz com que novos fluxos de usuários legítimos sejam descartados. Dessa forma, estudos presentes na literatura têm sido desenvolvidos com o objetivo de criar mecanismos de segurança para conter esses ataques [15, 44].

Scaranti *et al.* [45] propuseram um *IDS* baseado em clusterização *on-line*, aplicando o algoritmo *DenStream* para detectar ataques em redes *SDN*. Os autores exploram a capacidade da técnica não supervisionada para identificar ataques de *DDoS* e de *Portscan* simultâneos e de diferentes intensidades e durações. A avaliação do sistema proposto foi realizada por meio de dados emulados gerados pelos próprios autores. Além disso, o método utilizado foi comparado com um algoritmo de classificação *online* e os resultados obtidos demostraram que o método proposto obteve resultados de *f1-score* de 99,60%. Por outro lado, o trabalho não avaliou a mitigação dos ataques detectados, sendo uma atividade fundamental no processo de construção de *IDSs*

Carvalho *et al.* [46] apresentaram um *ecossistema* para auxiliar no gerenciamento da segurança das redes *SDN*. O *ecossistema* proposto pelos autores é composto de quatro módulos com funções bem definidas. O primeiro módulo realiza a coleta e armazenamento de registros de fluxo *IP*. O segundo módulo caracteriza o perfil normal da rede com base nos dados coletados e faz a detecção de eventos anômalos. Nesse módulo, a análise do comportamento do tráfego para a detecção de anomalias é realizada a cada 30 segundos. O terceiro módulo apresentado aplica políticas de mitigação, minimizando os efeitos causados por um ataque. Por fim, no quarto módulo, são gerados relatórios sobre os ataques detectados com o objetivo de validação e auditoria. A validação do sistema proposto foi feita utilizando fluxos *IP* emulados. De acordo com os resultados apresentados pelos autores, o *ecossistema* se mostrou eficiente nas fases de detecção e de mitigação de eventos anômalos.

Com o crescimento das aplicações de reconhecimento de imagem, processamento na-

tural de linguagem e bioinformática, os modelos de *Deep Learning* (*DL*) tiveram um papel fundamental na solução desses tipos de problema devido à sua grande capacidade de extrair conhecimento em larga escala de dados complexos, obtendo vantagens em seus resultados se comparados às técnicas tradicionais de *Machine Learning* [16, 17]. Em *cyber* segurança, os modelos de *DL* estão sendo aplicados em diversas áreas, por exemplo, detecção de intrusão [47], detecção de *malware* [48], detecção de *spam* [48] e detecção de ataques de *DDoS* no ambiente *SDN* [20, 49].

Tang *et al.* [50] empregaram *DL* para a detecção de fluxos anômalos em redes *SDN*. Os autores propuseram uma *Deep Neural Network* (*DNN*) para um sistema de detecção de intrusos e o modelo foi treinado utilizando o conjunto de dados NSL-KDD, o qual é composto de 41 atributos. No entanto, foi utilizado apenas um subconjunto de seis atributos. Uma vez que uma anomalia de rede foi detectada e identificada, utilizou-se o protocolo *OpenFlow* para modificar a tabela de fluxo, mitigando o ataque detectado. Por meio de experimentos, o modelo proposto obteve uma acurácia de apenas 75.75%. A pequena quantidade de atributos pode ter influenciado a acurácia baixa apresentada pelo sistema.

Li *et al.* [51] propuseram um mecanismo de detecção e defesa de ataques de *DDoS* em ambientes *SDN* empregando um modelo *ensemble* supervisionado de *Deep Learning*. Em sua construção, foram combinadas as arquiteturas de *Convolutional Neural Network* (*CNN*), *Long Short-Term Memory* (*LSTM*) e *Recurrent Neural Network* (*RNN*). Assim, após a detecção de um ataque, o controlador *SDN* gera políticas de descartes e as emite para os *switches*. Na condução dos testes, foi empregado o *dataset* ISCX para treinar o modelo de detecção e verificar o desempenho do mecanismo de defesa contra ataques de *DDoS*. De acordo com os resultados obtidos, o método de defesa apresentado teve uma taxa de acurácia de 98% na fase de teste. Priyadarshini e Barik [52] também exploraram a arquitetura de rede *LSTM* no desenvolvimento de um *framework* para a detecção e a mitigação de ataques de *DDoS*. Nas fases de treinamento e testes, foram utilizados subconjuntos dos *datasets* CTU-13 e ISCX 2012 IDS. Na fase de testes, o *framework* proposto obteve uma taxa de acurácia de 98%.

Shone *et al.* [47] propuseram um novo modelo *ensemble* de *DL* para *NIDS*, que combina *deep* e *shallow learning*. O modelo utilizou os métodos *Non-symmetric Deep Auto-Encoder* (*NDAE*) e *Random Forest*. O *NDAE* foi aplicado com o objetivo de minimizar a operação humana na seleção e na redução da dimensionalidade dos atributos utilizados para a detecção de anomalias. Porém, ele não é adequado para classificar e detectar anomalias. Neste sentido, os autores utilizaram a *Random Forest* na fase de detecção. Nos cenários de testes, foram utilizados os *datasets* KDD CUP 99 e NSL-KDD, cujas as taxas de acurácia obtidas foram de 97.85% e 80.58%, respectivamente.

Dey e Rahman [53] apresentaram um modelo de detecção de anomalias com base em fluxos *IP*. Esse modelo foi implementado no controlador *SDN* e utilizou redes neurais pro-

fundas, combinando duas arquiteturas: *Gated Recurrent Unit* e *Long Short-Term Memory* (*GRU-LSTM*). Os autores notaram que eliminando atributos irrelevantes e redundantes, era possível melhorar o desempenho do modelo. Nesse sentido, foram empregados dois métodos de seleção de atributos para cada tipo de anomalia analisada: a *ANOVA F-Test* e a *Recursive Feature Elimination*. Na realização dos experimentos, foi utilizado o conjunto de dados NSL-KDD e os resultados experimentais demonstraram uma acurácia de 87%.

Haider *et al.* [21] propuseram um *framework* para a detecção de ataques de *DDoS* empregando uma arquitetura *ensamble* de *Convolutional Neural Network* (*CNN*) supervisionada. Esse *framework* combina as saídas de duas redes *CNNs* para a tomada de decisão, detectando se o tráfego de rede era normal ou se havia um ataque de *DDoS*. A estratégia adotada pelos autores foi acoplar o *framework* como parte integrante do plano de controle. A avaliação do *framework* foi realizada utilizando o conjunto de dados CICIDS-2017. Os resultados obtidos demonstram uma acurácia de detecção de aproximadamente 99%.

Yang *et al.* [54] também projetaram um *framework* com o objetivo de detectar ataques de *DDoS*, utilizando uma abordagem não supervisionada de *Auto-Encoder* (*AE*). Os autores utilizaram o *AE* para criar uma assinatura do comportamento normal da rede com base em dados históricos. O *framework* proposto é composto de dois módulos principais: extração de atributos e detecção *online*. Na extração de atributos, o módulo realiza a coleta dos atributos presentes nas tabelas de fluxos dos *switches* e os exporta para o módulo de detecção. Na detecção, o *AE* gera uma assinatura com base nos atributos de fluxos exportados utilizando uma janela temporal de 10 segundos. Um ataque é detectado caso o tráfego real ultrapasse um limiar predeterminado da assinatura gerada. Os *datasets* utilizados para treinamento e teste foram o UNB 2017 e o MAWI. Em ambos os *datasets*, a taxa de acurácia obtida foi de aproximadamente 95%.

Liu *et al.* [55] propuseram um método de detecção de ataques de *DDoS* em dois níveis baseado em entropia e *Deep Learning*. Primeiro, o mecanismo de entropia detecta componentes e portas suspeitos. Em seguida, foi utilizada uma arquitetura de *CNN* para distinguir o tráfego normal do tráfego suspeito. E por fim, o controlador executa a estratégia de defesa para interceptar o ataque. Foram realizados testes experimentais utilizando o conjunto de dados CICDDoS 2019 e os resultados dos experimentos indicaram uma taxa de acurácia de aproximadamente 98%

Assis *et al.* [22] apresentaram um sistema modular de defesa *SDN* baseado na análise de fluxos IP, que utilizou o método de aprendizagem profunda *GRU* para detecção de intrusões. O modelo proposto foi avaliado e comparado com diferentes abordagens de aprendizagem de máquina em dois conjuntos de dados públicos, o CICDDoS 2019 e o CICIDS 2018. Em ambos os cenários avaliados, o método proposto apresentou resultados médios superiores a 97% com relação às *precision, recall, f1-sore*.

Lent *et al.* [56] propuseram um sistema de detecção e mitigação de anomalias em redes *SDN* empregando a *GRU* em conjunto com a lógica Fuzzy. A *GRU* foi utilizada para prever o comportamento normal da rede, com base na análise de fluxos *IP*, e a lógica Fuzzy foi empregada para a detecção dos eventos anômalos. O sistema desenvolvido foi testado em dois conjuntos de dados: tráfego de rede emulado e o conjunto público de dados CICDDoS 2019. Durante a fase de avaliação, o sistema proposto foi comparado com outros métodos de *Deep Learning*: *DNN*, *CNN* e *LSTM*. No conjunto de dados emulados e nos dados CICDDoS 2019, o sistema obteve *F1-score* de 98.9% e 93.2%, respectivamente.

2.3 Considerações sobre o capítulo

Neste capítulo, foram apresentados sistemas que atuam na detecção e na mitigação de anomalias em redes *SDN*, principalmente ataques de *DDoS*. Os trabalhos analisados exploram diferentes técnicas para detectar esses ataques, as quais focam, em sua maioria, em algoritmos tradicionais de *Machine Learning* e nos modelos de *Deep Learning*. A capacidade de generalização e extração de conhecimento dos modelos de *Deep Learning* fez com que os sistemas que utilizaram essas técnicas obtivessem resultados superiores se comparados aos algoritmos tradicionais de *Machine Learning*. Outra característica das soluções apresentadas é o plano de rede em que foram implementadas. A maioria dos trabalhos implementaram as soluções no plano de aplicação para não comprometer as atividades operacionais do plano de controle, pois, durante um ataque em um ambiente real, a quantidade de mensagens trocadas entre o controlador e os dispositivos de rede aumenta consideravelmente, podendo sobrecarregá-lo e torná-lo indisponível.

Com relação aos trabalhos presentes na literatura, diversos sistemas de detecção de anomalias avaliam apenas alguns tipos de ataque de *DDoS* [2, 14, 17, 57]. Diferentemente desses trabalhos, uma das principais contribuições apresentadas pelos sistemas propostos nesta tese é a detecção de 12 tipos de ataque de *DDoS*: *NTP*, *DNS*, *LDAP*, *MSSQL*, *NetBIOS*, *SNMP*, *SSDP*, *UDP*, *UDP-Lag*, *Web-DDoS* (*ARME*), *SYN* e *TFTP*. O sistema proposto é capaz de aprender o comportamento normal do tráfego de rede, o que é uma vantagem para detectar ataques *zero-day*. Além disso, foi empregado o *framework GAN* que fornece uma taxa de detecção mais precisa e menos sensível a ataques adversários

Tabela 1 – Trabalhos relacionados e analisados.

Autor	**Alvo**	**Algoritmo/ Técnica**	**Plano**	**Dataset**
Hamamoto *et al.* [36]	Detecção	*GA* Lógica Fuzzy	Aplicação	Tráfego real
Assis *et al.* [37]	Detecção e mitigação	*Holt-winters* GA Lógica Fuzzy Teoria dos jogos	Aplicação	Tráfego real
Zerbini *et al.* [38]	Detecção e mitigação	Wavelet *K-means* *Random Forest*	Aplicação Controle	Emulado
Jmila *et al.* [39]	Detecção	*Adaboost* *Bagging* *Gradientboosting* *Logistic Regression* *Decision Tree* *Random Forest* *Support Vector Classifier*	Aplicação	NSL-KDD UNSW-NB 15
Kasim *et al.* [40]	Detecção	*CNN* *LSTM*	Aplicação	CICIDS
Pingale *et al.* [41]	Detecção	*CNN* *Auto Encoder*	Aplicação	NSL-KDD CI-CIDS2018 UNSW-NB15
Zhu *et al.* [42]	Detecção	*GAN* *GRU*	Aplicação	Kitsune CI-CIDS2017 MAWILA UNSW-NB15

Tabela 1 – Trabalhos relacionados e analisados.

Autor	**Alvo**	**Algoritmo/ Técnica**	**Plano**	**Dataset**
Scaranti *et al.* [45]	Detecção	*DenStream*	Aplicação	Emulado
Carvalho *et al.* [46]	Detecção e mitigação	*ACO* Regressão Logística	Aplicação	Emulado
Tang *et al.* [50]	Detecção e mitigação	*DNN*	Controle	NSL-KDD
Li *et al.* [51]	Detecção e mitigação	*CNN* *LSTM* *RNN*	Aplicação	ISCX IDS 2012
Priyadarshini e Barik [52]	Detecção e mitigação	*LSTM*	Aplicação	CTU-13 ISCX IDS 2012
Shone *et al.* [47]	Detecção	*NDAE* *Random Forest*	Aplicação	KDD Cup 99 NSL-KDD
Dey e Rahman [53]	Detecção	*GRU* *LSTM*	Controle	NSL-KDD
Haider *et al.* [21]	Detecção	*CNN*	Controle	CICIDS-2017
Yang *et al.* [54]	Detecção	*AE*	Controle	UNB 2017 MAWI
Liu *et al.* [55]	Detecção	*CNN*	Aplicação	CICDDoS 2019
Assis *et al.* [22]	Detecção e mitigação	*GRU*	Aplicação	CICIDS 2018 CICDDoS 2019
Lent *et al.* [56]	Detecção e mitigação	*CNN* Fuzzy	Aplicação	Emulado CICDDoS 2019

Fonte: O próprio autor.

3 Fundamentação teórica

3.1 Detecção de Intrusão e Anomalias

Os sistemas que utilizam redes de computadores sofrem ataques e invasões frequentemente. Mecanismos de segurança são utilizados para monitorar, analisar e detectar ações de agentes maliciosos na rede ou em um *host*. Para esse propósito, é comum a utilização de sistemas de detecção de intrusão (*IDS*, do inglês *Intrusion Detection System*) [58, 59]. O objetivo dos *IDS* é garantir as políticas de segurança, ou seja, confiabilidade, integridade e disponibilidade. Os *IDS* podem ser classificados em três categorias: *Host-based* (*Host-based Intrusion Detection System - HIDS*), *Network-based* (*Network Intrusion Detection System - NIDS*) e uma versão híbrida, que combina ambas as abordagens [60, 61].

O *HIDS* realiza o monitoramento sobre as atividades de um único sistema ou computador, coletando informações de arquivos de *logs* do sistema que está sendo monitorado. Esses arquivos podem conter informações úteis sobre os eventos ocorridos, permitindo detectar ataques e atividades suspeitas por parte dos usuários. As informações são registradas por um mecanismo chamado de trilha de auditoria, em que é possível identificar qual processo iniciou um evento e, por consequência, os possíveis usuários associados a esse evento.

Os *NIDS* são ferramentas essenciais para garantir as políticas de segurança e auxiliar o administrador de rede nas atividades de gerenciamento e monitoramento. Eles são aplicados na detecção de eventos anômalos que podem ocorrer em uma rede. Um *NIDS* monitora o tráfego, inspecionando os campos do cabeçalho dos pacotes ou aplicando protocolos de gerenciamento. Eles são classificados de acordo com as abordagens adotadas no processo de detecção, as quais podem ser divididas em duas categorias: baseada em padrões de assinatura e baseada em anomalias [15, 23].

Na abordagem baseada em padrões de assinaturas, é utilizada uma base contendo as características dos ataques conhecidos. O sistema monitora continuamente as atividades da rede. Durante o processo de monitoramento, um ataque é detectado quando há uma correspondência entre os padrões armazenados e o comportamento atual da rede. Nessa abordagem, serão identificados apenas os eventos anômalos que se encontram registrados na base de assinaturas, ou seja, os ataques já conhecidos. Por outro lado, na abordagem baseada em anomalias, por meio dos registros históricos do tráfego da rede, é caracterizado um perfil como sendo normal. Desse modo, o sistema foca em analisar o tráfego e definir limiares máximos e mínimos de determinados atributos da rede. Uma eventual anomalia é detectada pelo sistema quando, em um instante de análise do segmento de rede, os limiares de normalidades predefinidos são ultrapassados [56, 62].

As características intrínsecas de cada abordagem definem suas vantagens e desvantagens. As principais vantagens da abordagem baseada em padrões de assinatura são a detecção eficiente dos principais ataques conhecidos e a diminuição da taxa de falsos-positivos. Uma desvantagem apresentada por ela é a constante atualização dos padrões de assinaturas, pois só serão detectadas as anomalias que estiverem registradas. Outra desvantagem é o tamanho da base de assinaturas, que pode ser muito grande, afetando o desempenho do sistema durante o processo de monitoramento.

A principal vantagem apresentada pelas abordagens baseadas em anomalias é a detecção de eventos anômalos desconhecidos e ataques do tipo *zero-day*, pois o perfil é caracterizado com base no comportamento normal das atividades da rede. Dessa maneira, a detecção é realizada sem a necessidade de conhecimentos prévios sobre os comportamentos das anomalias. Porém, a modelagem do comportamento normal é complexa e requer o emprego de técnicas capazes de lidar com as constantes flutuações de normalidade apresentadas pelo tráfego de rede. Devido a essa complexidade, a abordagem baseada em anomalias apresenta como desvantagem a elevada taxa de falsos-positivos, em que o tráfego de usuários legítimos pode ser detectado como anômalo [62].

A combinação entre um *HIDS* e um *NIDS* dá origem a uma abordagem híbrida de sistema de detecção de intruso, em que o sistema realiza o monitoramento das atividades locais e também monitora a operação da rede. Assim, o *IDS* detecta as atividades realizadas por agentes maliciosos contra o *host* e contra a rede monitorada.

3.1.1 Anomalias em redes

Em redes de computadores, as anomalias podem ser definidas como eventos que ocorrem na rede causando comportamentos diferentes do padrão normal. Um aspecto importante na detecção de anomalias é a identificação da sua natureza [15]. O contexto em que um possível evento anômalo é identificado, ou a forma como ele ocorreu, pode ser ou não considerado uma anomalia. Esse aspecto é essencial para direcionar o modo como o sistema vai lidar com o evento ocorrido. Com base na sua natureza, uma anomalia pode ser categorizada das seguintes maneiras: anomalia pontual, anomalia contextual e anomalia coletiva [63, 64].

Uma anomalia pontual ocorre quando determinada instância de dados individual se desvia do padrão usual de um conjunto de dados. Um exemplo prático dessa natureza é uma aplicação de rede que tem certo consumo médio de largura de banda e, em dado instante, esse serviço passa a consumir o dobro. Essa situação pode ser considerada uma anormalidade pontual.

A anomalia contextual, também chamada de condicional, ocorre quando uma instância de dados se comporta de maneira anômala dependendo do contexto em que se encontra. Por exemplo, a rede de uma instituição de ensino, em dias letivos, tem uma taxa média

de envio e recebimento de *bits* e pacotes. Porém, em determinado contexto, como um feriado, essas taxas serão menores devido à ausência de atividades na instituição naquele dia. Embora tenha ocorrido um decréscimo nessas taxas, esse comportamento pode ser considerado normal. Já em um dia útil, esse comportamento pode ser considerado uma anomalia contextual.

A anomalia coletiva ocorre quando uma coleção de instâncias de dados semelhantes se comporta de maneira anômala em relação a todo o conjunto de dados. Considere a seguinte sequência de ação realizada em um computador: "*HTTP-web, buffer-overflow, HTTP-web, HTTP-web, FTP, HTTP-web, SSH, HTTP-web, SSH, buffer-overflow*". Essa sequência representa um possível ataque *web-based* realizado por uma máquina remota seguida pela cópia de dados do computador *host* para um destino remoto via *FTP*. Contudo, se esses eventos ocorrerem em uma sequência diferente podem não indicar uma anormalidade [65].

Os eventos anômalos podem ser classificados de acordo com a circunstância ocorrida. Existem anomalias referentes a problemas de falhas e desempenho, ou seja, a causa do evento originado não está relacionada à presença de agentes maliciosos, e existem anomalias referentes a problemas de segurança, isto é, o evento originado foi realizado por um agente malicioso com o propósito de violar as políticas de segurança da rede [66].

Anomalias causadas por falhas e desempenho podem ocorrer, por exemplo, em falhas ou má configuração dos equipamentos, quando um servidor recebe inúmeras solicitações de usuários legítimos ou o limite da largura de banda é atingido, causando congestionamento no tráfego. A ocorrência de *flash crowd*, *babbling node*, tempestade de *broadcast*, *bugs* no *software* de roteamento e erros de configuração são os principais eventos que provocam esse tipo de anomalia.

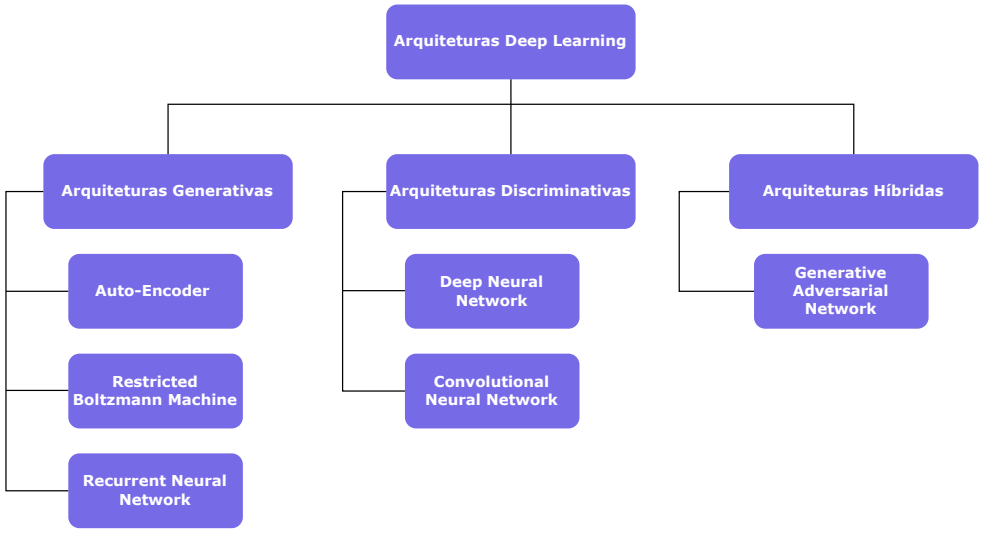
Por outro lado, anomalias causadas por questões de segurança estão relacionadas à presença de agentes maliciosos sobrecarregando o tráfego da rede. Essa situação ocorre quando o agente sobrecarrega determinado serviço com o objetivo de torná-lo indisponível ou em situações nas quais o agente busca por portas abertas no sistema a fim de fazer invasões. Alguns exemplos disso são os ataques de negação de serviço (*Denial of Service - DoS*), o escaneamento de portas (*Portscan*) e o *Worm*.

3.2 Redes neurais profundas

Deep Learning (*DL*) é uma subárea de *Machine Learning* (*ML*) que representa uma evolução nas técnicas de redes neurais artificiais (RNA). Uma RNA composta de diversas camadas é denominada rede neural profunda (*DNN*, do inglês *Deep Neural Network*). A aplicação dos algoritmos de *DL* se tornou uma tendência devido à capacidade de abstração e generalização no processo de aprendizagem em diversos domínios, principalmente para modelar conceitos e relacionamentos complexos [67]. Os algoritmos de *DL* forne-

cem arquiteturas profundas formadas por várias camadas de unidades de processamento, permitindo a extração hierárquica de informações de dados brutos.

As arquiteturas de redes neurais profundas têm sido aplicadas em diversas áreas do conhecimento, por exemplo, processamento de áudio [68, 69, 70], sistemas autônomos [71, 72, 73], cibersegurança [74, 75, 76], reconhecimento de imagem e vídeo [77, 78, 79] e processamento de linguagem natural [80, 81, 82]. Nessas áreas, os algoritmos *DL* mostraram desempenho superior ao dos algoritmos clássicos de *ML* relacionados às tarefas de classificação, redução de dimensionalidade e extração automática de atributos. Diferentemente dos algoritmos tradicionais de *ML*, os de *DL* têm a capacidade de realizar simultaneamente o aprendizado dos atributos (*feature-learning*) e as tarefas de classificação ou *clusterização*, além de encontrar correlações entre os dados utilizados no processo de aprendizagem. As arquiteturas de *DL* podem ser divididas em três subcategorias: generativa, discriminativa e híbrida [57]. A Figura 1 sumariza como elas estão organizadas.



Fonte: O próprio autor.

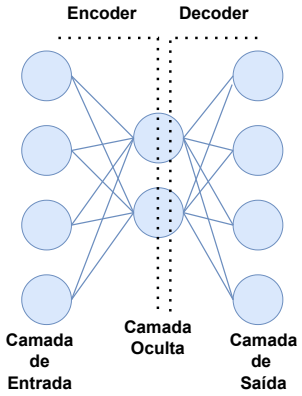
Figura 1 – Arquiteturas de *Deep Learning*.

3.2.1 Arquiteturas generativas

Arquiteturas generativas ou aprendizagem não supervisionada são redes neurais profundas que têm a capacidade de extrair o conhecimento automaticamente, utilizando dados brutos não rotulados. As arquiteturas de rede a seguir podem ser consideradas as principais nessa categoria:

- *Auto-Encoder* (*AE*): *AE* é uma arquitetura de rede neural profunda que foi proposta por Hinton *et al.* [83] e é utilizada principalmente para a redução de dados brutos de alta dimensionalidade, produzindo uma representação similar menor. Essa redução facilita as atividades de classificação, visualização e armazenamento de dados de alta

dimensionalidade. A organização arquitetural do *AE* é composta de uma camada de entrada, uma camada oculta e uma camada de saída. A quantidade de neurônios das camadas de entrada e de saída é igual ao tamanho do vetor de atributos brutos. O *AE*, em sua construção, combina duas estruturas: um codificador (*encoder*) e um decodificador (*decoder*), conforme ilustrado na Figura 2. O *encoder* é responsável por extrair os atributos e aprender uma representação de baixa dimensionalidade dos dados de entrada. Por outro lado, o *decoder* recebe os dados de baixa dimensionalidade e reconstrói os atributos originais. Além disso, podem ser empregadas diversas camadas ocultas, formando uma rede neural profunda. Essa estrutura é chamada de *Stacked Auto-Encoder* (*SAE*) [84]. A utilização de diversas camadas ocultas permite o aprendizado de maneira progressiva e hierárquica, fornecendo uma representação mais precisa dos atributos transformados. Outra extensão do *AE* é o *Denoising Auto-Encoder* (*DAE*), aplicado na eliminação de ruídos de dados contaminados para produzir uma representação refinada dos atributos empregados.

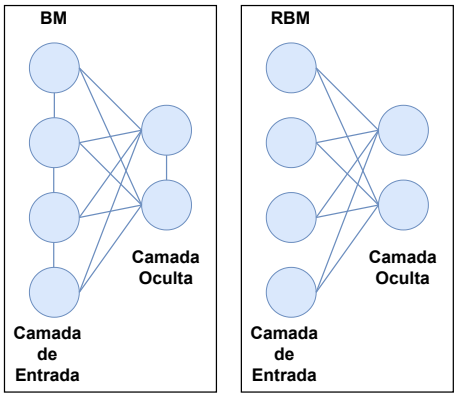


Fonte: O próprio autor.

Figura 2 – Arquitetura conceitual de um *Auto-Encoder*.

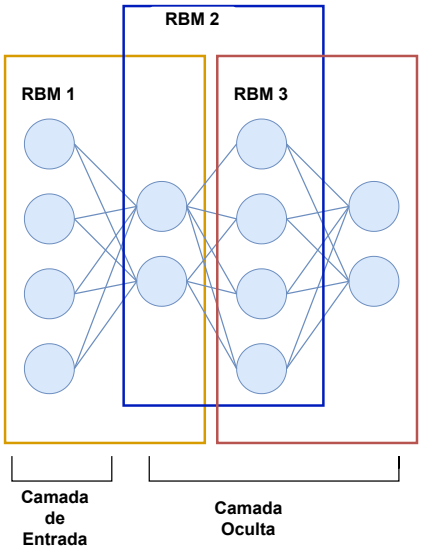
- *Restricted Boltzmann Machine* (*RBM*): introduzidas pelos autores Hinton e Sejnowsk [85], as *Boltzmann Machines* (*BM*) são redes neurais estocásticas e generativas capazes de aprender representações internas e resolver problemas combinatórios difíceis. Uma rede *BM* é composta de diversas unidades de processamento, sendo que cada uma é conectada a todas as outras. Não há saída única na máquina, apenas unidades ocultas e unidades visíveis, onde as unidades visíveis representam os dados. Esse modelo tenta entender a distribuição dos dados e recriá-los com base nessa distribuição. Mesmo com apenas algumas unidades, há muitas conexões em uma *BM*, resultando em um processo de aprendizagem lento. Para resolver o problema relacionado à complexidade das conexões, foi proposta a *Restricted Boltzmann machine* (*RBM*) [86]. Na arquitetura da *RBM*, as conexões entre as unidades de processamento da mesma camada são eliminadas. A *RBM* é composta de uma camada visível e uma oculta. A visível representa os atributos de entrada, e cada

unidade de processamento dessa camada é conectada às unidades da camada oculta, associadas com um valor de ponderação. As unidades de processamento da camada oculta aprendem a distribuição dos atributos das variáveis de entrada. A Figura 3 ilustra a diferença arquitetural entre as redes *BM* e *RBM*. A Figura 4 ilustra a utilização de duas ou mais *BM*, chamada de *Deep Boltzmann machine* (*DBM*) [87].



Fonte: O próprio autor.

Figura 3 – Diferença entre as arquiteturas *BM* e *RBM*.

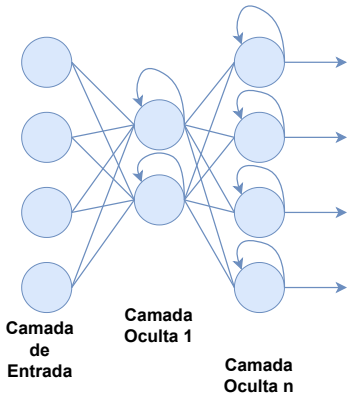


Fonte: O próprio autor.

Figura 4 – Arquitetura conceitual de uma *DBM*.

- *Recurrent Neural Network* (*RNN*): proposta por Hopfield [88], a *RNN* é uma arquitetura de rede neural que emprega a retroalimentação no processo de aprendizagem, isto é, o sinal de saída da etapa anterior é utilizado como entrada da etapa atual, conforme ilustrado na Figura 5. Esse tipo de arquitetura de rede distingue-se por sua capacidade de extrair conhecimento de dados organizados de maneira sequencial. A *RNN* é aplicada principalmente em problemas nos quais cada amostra depende da análise das amostras anteriores, por exemplo, geração automática de texto, reconhecimento de fala, dados de sensores, séries temporais etc. As *RNNs* foram estendidas

com diferentes variantes de unidades de memória, incluindo a *Long Short-Term Memory* (*LSTM*) [89] e a *Gated Recurrent Unit* (*GRU*) [90].



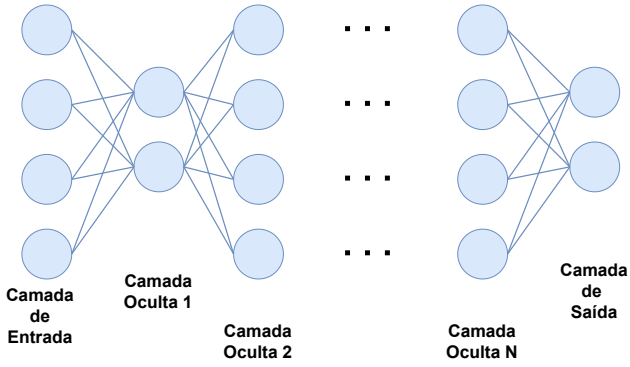
Fonte: O próprio autor.

Figura 5 – Arquitetura conceitual de uma *RNN*.

3.2.2 Arquiteturas discriminativas

As arquiteturas discriminativas ou de aprendizagem supervisionada são utilizadas para classificar os dados de entrada, isto é, indicados os atributos de uma instância, elas preveem um rótulo ou uma classe à qual esses dados pertencem. As principais arquiteturas discriminativas são as seguintes:

- *Deep Neural Network* (*DNN*): é uma arquitetura de rede neural multicamada, composta de diversas camadas e neurônios entre a camada de entrada e a de saída [91], conforme ilustrado na Figura 6. *DNN* é uma rede do tipo *feedforward*, em que os atributos de entrada fluem da camada de entrada para a de saída sem retroalimentação.

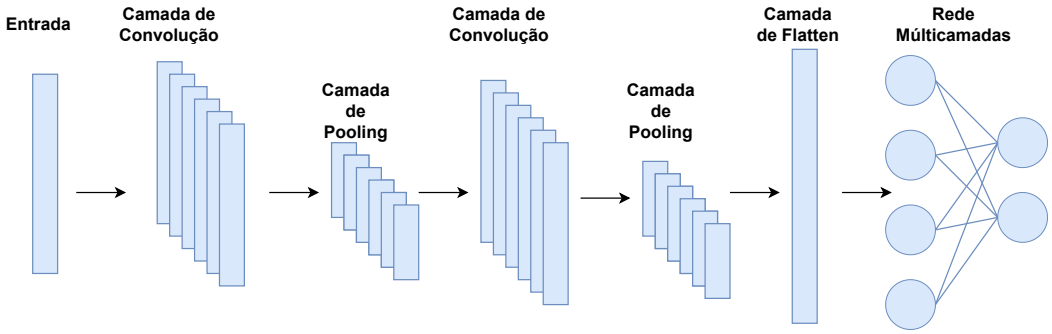


Fonte: O próprio autor.

Figura 6 – Arquitetura conceitual de uma *DNN*.

- *Convolutional Neural Network* (*CNN*): o conceito de redes neurais convolucionais foi introduzido por LeCun *et al.* [92]. As *CNNs* treinam várias camadas com

mapeamentos não lineares para classificar dados de alta dimensionalidade em um conjunto de classes de saída. A principal vantagem da *CNN* é a capacidade de extrair de maneira automática as características (*feature-learning*) de dados brutos. A arquitetura de uma rede *CNN* é formada por camadas de convolução e *pooling*. Na camada de convolução, são empregados filtros de convolução para produzir mapas de características. Já a camada de *pooling* é utilizada para reduzir a dimensionalidade desses mapas. Assim, a dimensionalidade será reduzida, mas serão selecionadas as principais características. Os mapas de características encontrados podem ser utilizados como entrada para uma rede multicamadas, completando o processo de aprendizagem de um problema de classificação. A Figura 7 ilustra uma arquitetura conceitual de uma rede *CNN*.

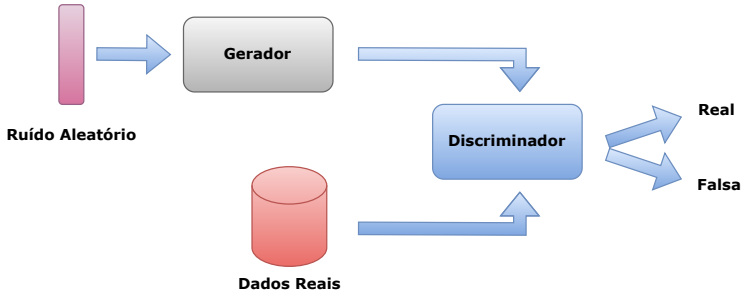


Fonte: O próprio autor.

Figura 7 – Arquitetura conceitual de uma *CNN*.

3.2.3 Arquiteturas híbridas

Generative Adversarial Network (*GAN*) [32] é um *framework* para a estimativa de modelos generativos por meio de um processo contraditório, em que dois modelos, um discriminador e um gerador, são treinados simultaneamente. A *GAN* conta com um jogo de *minimax*, no qual uma rede busca maximizar o valor da função e a outra tenta minimizá-lo. A cada rodada do treinamento contraditório, por meio de ruídos aleatórios, o gerador produz amostras sintéticas, que são apresentadas ao discriminador juntos às amostras originais. A função do discriminador é diferenciar as amostras originais das produzidas pelo gerador. Por outro lado, o treinamento do gerador é efetivo quando as amostras produzidas são similares às amostras originais. O diagrama apresentado na Figura 8 ilustra uma arquitetura conceitual do *framework GAN*.



Fonte: O próprio autor.

Figura 8 – Arquitetura conceitual do *framework GAN*.

3.2.4 Ataques adversários contra redes neurais profundas

Estudos presentes da literatura demostram que as arquiteturas de rede neurais profundas são vulneráveis a exemplos adversários (*adversarial exemples*), ou seja, ações executadas por agentes maliciosos fazendo com que os métodos de *DL* cometam erros na detecção de ataques [28, 93, 94]. Os ataques adversários podem ser amplamente definidos como uma classe de ataques que visam enganar um método de *DL* inserindo exemplos adversários na fase de treinamento, conhecida como *poisoning attack* [95, 96], ou na fase de inferência, chamada de *evasion attack* [97, 98]. Ambos os ataques diminuirão significativamente a robustez dos métodos de DL, causando problemas de segurança.

- *Poisoning Attack*: os atacantes podem acessar e modificar o conjunto de dados de treinamento para impactar os modelos treinados [99, 100]. A maneira como os atacantes injetam as amostras falsas nos dados de treinamento para gerar um modelo impreciso pode ser vista como "envenenamento". Esse tipo de *adversarial attack* geralmente leva a uma diminuição da acurácia [101] ou à classificação incorreta das amostras recebidas pelos métodos durante a fase de operação [102].

- *Evasion attack*: esse ataque consiste em modificar cuidadosamente os atributos das amostras de entrada na fase de operação da rede para que elas sejam classificadas erroneamente, ou seja, para que amostras anômalas sejam classificadas como normais [97, 103]. Dessa forma, a acessibilidade do conjunto de dados de treinamento não é mais necessária, uma vez que os atacantes, exploram as vulnerabilidades dos modelos modificando os dados de teste para produzir o erro desejado [104].

3.3 Hiperparâmetros

Em redes neurais profundas, os hiperparâmetros são parâmetros que não são aprendidos pelo modelo durante o treinamento, mas precisam ser definidos antes do início do processo de treinamento. Esses hiperparâmetros controlam o comportamento e o desempenho da rede neural durante o treinamento e a inferência.

3.3.1 Número de camadas ocultas

O número de camadas ocultas em uma rede neural também é chamado de *depth* (profundidade). O termo "*deep*", em *Deep learning*, refere-se à quantidade de camadas ocultas, isto é, à profundidade de uma rede neural. Ao projetar uma rede neural profunda, o número de camadas determina a capacidade de aprendizagem da rede. Para aprender e generalizar os padrões não lineares de um conjunto de dados, a rede neural deve ser composta de uma quantidade suficiente de camadas ocultas. Quando a quantidade de amostras e a complexidade do conjunto de dados aumentam, aumenta também a capacidade de aprendizagem necessária para uma rede neural [105].

Um número muito pequeno de camadas ocultas gera uma rede menor, que consequentemente não aprenderá os padrões complexos presentes nos dados de treinamento e, durante a fase de operação da rede, não terá um bom desempenho com os novos dados que serão apresentados, gerando um subajuste (*underfitting*). Por outro lado, muitas camadas ocultas vão gerar uma rede maior, que pode sobreajustar (*overfitting*) os dados de treinamento, ou seja, ajusta-se muito bem aos dados apresentados durante a fase de treinamento, memorizando a distribuição em vez de generalizar e extrair os padrões desses dados.

3.3.2 Número de neurônios

O número de neurônios em uma rede neural também é chamado de largura. As camadas ocultas são compostas de neurônios, sendo outro fator que afeta a capacidade de aprendizagem da rede. Muitos neurônios criam camadas ocultas grandes, que podem sobreajustar os dados de treinamento. Já um número muito pequeno pode subajustar os dados de treinamento.

Quando o número de camadas ocultas e de neurônios aumenta, a rede se torna muito grande e o número de parâmetros aumenta significativamente. Para treinar redes tão grandes, muitos recursos computacionais são necessários. Assim, grandes redes neurais são caras em termos de recursos computacionais.

3.3.3 Funções de ativação

As funções de ativação entre as camadas de uma rede são utilizadas para introduzir uma não linearidade, sendo fundamental para aprender padrões complexos. O desempenho de um modelo de rede neural varia significativamente dependendo do tipo de função de ativação que usamos dentro das camadas ocultas. As principias funções de ativação utilizadas entre as camadas ocultas são:

- Sigmoide.
- Tangente Hiperbólica.

- ReLU.

Na camada de saída, também é utilizada uma função de ativação, que depende do tipo de problema que está sendo resolvido. Nessa camada, as principais funções empregadas são as indicadas a seguir.

- Linear: utilizada em problemas de regressão com um neurônio na camada de saída.
- Sigmoide: utilizada em problemas de classificação binária com um neurônio na camada de saída.
- Softmax: utilizada em problemas de multiclassificação com um neurônio por classe na camada de saída.

3.3.4 Tipos de algoritmo de otimização

Os algoritmos de otimização, também chamados de *optimizer*, são utilizados para minimizar a função de perda (*function loss*), atualizando os pesos sinápticos da rede. A descida do gradiente (*Gradient Descent*) foi o primeiro algoritmo desenvolvido para otimização utilizado para calcular o gradiente da função de perda. Ao longo dos anos, outros tipos de otimizadores foram desenvolvidos para lidar com as deficiências do algoritmo de descida de gradiente, sendo eles Adam [106], Adagrad [107], Adadelta [108], Adamax[109], Nadam [110] e RMSProp [111].

3.3.5 Taxa de aprendizagem

Durante a otimização, o otimizador dá pequenos passos para descer a curva de erro. A taxa de aprendizagem refere-se ao tamanho do passo. Ele determina o quão rápido ou lento o otimizador desce a curva de erro. Um valor maior de taxa de aprendizagem pode ser usado para treinar a rede mais rapidamente. Porém, um valor muito grande fará com que a função de perda oscile em torno do mínimo e nunca encontre o ponto de convergência. O valor padrão empregado pela maioria dos otimizadores é uma pequena taxa de aprendizagem com valor igual a 0,001, o qual pode ser aumentado sistematicamente se a rede demorar para convergir [112].

3.3.6 Tipos de função de perda

Durante o treinamento da rede, deve haver uma maneira de medir o seu desempenho, ou seja, verificar como ela está aprendendo a distribuição dos dados presentes no conjunto de treinamento. A função de perda calcula o erro entre os valores previstos e os valores reais. O objetivo desse processo é minimizar a função de perda usando um dos algoritmos de otimização. A otimização do erro é realizada ao longo das épocas de treinamento. O

tipo de função de perda a ser utilizado durante o treinamento depende do problema que está sendo resolvido. As principais funções de perdas estão indicadas a seguir.

- *Mean Squared Error* (MSE), *Mean Absolute Error* (MAE), *Root Mean Square Error* (RMSE), Mean Absolute Percentage Error (MAPE) e *Huber Loss*: usadas para medir o desempenho em problemas de regressão.
- *Binary Cross-entropy*: usada para medir o desempenho em problema de classificação binária.
- *Categorical Cross-entropy*: usada para medir o desempenho em problemas de classificação multiclasse.
- *Sparse Categorical Cross-entropy*: usada para converter automaticamente rótulos de valor escalar em um vetor único em problemas de classificação multiclasse.

3.3.7 Tamanho do *batch*

O tamanho do *batch* (*batch size*) é o número de amostras usadas por atualização de gradiente. Os valores tipicamente utilizados são 16, 32, 64, 128, 256, 512 e 1024 [112]. Um tamanho de *batch* maior geralmente requer mais recursos computacionais por época, mas requer menos épocas para convergir. Por outro lado, um tamanho menor não requer grandes recursos computacionais, mas aumenta o número de épocas para convergir.

3.3.8 Número de épocas

O número de épocas é a quantidade de vezes que o modelo vê todo o conjunto de dados durante a fase de treinamento. Ele pode aumentar quando a rede treinada utiliza uma taxa de aprendizagem pequena e/ou o tamanho de *batch* é muito pequeno.

Um número muito grande de épocas pode causar um *overfitting* no conjunto de treinamento, ou seja, o erro da validação (amostras não apresentadas no treinamento) começa a aumentar e o erro do treinamento diminui ainda mais. Nessa situação, o modelo funciona bem nos dados de treinamento, mas reduz a sua capacidade de generalização para novas amostras.

3.3.9 Taxa de *dropout*

Dropout é um método de regularização eficaz para reduzir o *overfitting* e melhorar o erro de generalização em redes neurais profundas [113]. Durante o treinamento, as saídas de alguns neurônios de uma camada são ignoradas, removendo temporariamente da rede todas as suas conexões. A escolha da quantidade de neurônios removidos é aleatória com uma probabilidade fixa, determinada como *dropout rate*, com os valores variando de zero

a um. O valor zero indica que nenhum neurônio de uma camada será ignorado e o valor um remove todos os neurônios (não praticável).

3.4 Redes definidas por software

A evolução das redes de computadores levou à formação de estruturas dinâmicas e complexas. Em virtude dessa complexidade, a configuração e o gerenciamento dessa estrutura se tornaram uma atividade desafiadora. As redes são compostas por um grande número de dispositivos de rede, como *switches*, roteadores e *ppliances* de rede (*firewalls*, sistemas de detecção de intrusão, etc.), cada um deles implementando protocolos complexos. A tarefa de configurar todos os elementos da estrutura de rede para atender às diversas políticas de alto nível e responder aos inúmeros eventos de rede que podem ocorrer recai sobre o administrador de rede. No entanto, a configuração de rede ainda é uma tarefa difícil, uma vez que satisfazer as especificações das políticas de alto nível exige a implementação distribuída de configurações em termos de detalhes de baixo nível [4].

Redes definidas por software (*Software-Defined Networking – SDN*) introduziu ferramentas destinadas a simplificar a configuração e a gestão, bem como a possibilitar uma inovação mais significativa nas redes de comunicação. A característica central da arquitetura *SDN* reside na separação entre o plano de controle e o plano de dados, ou seja, o plano de controle é dissociado dos dispositivos de rede e centralizado em um controlador [5]. A centralização do plano de controle oferece uma visão abrangente da topologia de rede e permite a manipulação do fluxo de tráfego em tempo de execução, por meio de uma interface de software aberta e bem definida para o gerenciamento do plano de dados.

Neste seção, serão discutidos alguns conceitos fundamentais acerca das redes *SDN*. Serão abordados os detalhes dos componentes presentes nessa arquitetura, detalhando os planos de controle e de dados.

3.4.1 Arquitetura

A característica mais representativa em relação às redes *SDN* é o desacoplamento das tarefas de controle de rede e encaminhamento de pacotes. Isso basicamente se refere à migração de toda a inteligência de rede, que originalmente residia na infraestrutura de hardware, para uma entidade lógica centralizada baseada em software, enquanto todos os dispositivos de encaminhamento se tornam elementos simples de encaminhamento de pacotes. O desacoplamento dos planos de controle e dados na *SDN* implica na centralização lógica do controle e gerenciamento de todos os dispositivos de encaminhamento [114].

A arquitetura das redes *SDN* pode ser definida em quatro principais concepções [5]:

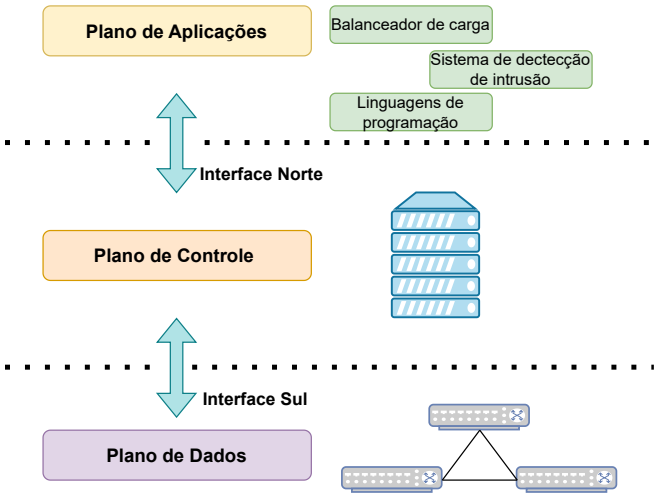
1. A separação entre o plano de controle e o plano de dados. Os *switches* passam a ter a função simples de encaminhamento de pacotes.

2. A lógica de encaminhamento é baseada em fluxos, sendo um fluxo uma sequências de pacotes entre um dispositivo de origem e um dispositivo de destino.

3. O controle desacoplado dos *switches* passar a ser centralizado unicamente em um controlador. O controlador é uma abstração de *software* que fornece recursos para a programação dos dispositivos de encaminhamento, provisionando uma visão global e centralizada de toda a topologia de rede.

4. A principal proposta de valor da rede *SDN* é a programação da rede através de software, facilitando as rotinas de gerência e controle da rede em sua totalidade.

A estrutura das redes *SDN* consiste em três partes principais. No nível mais baixo, inclui o plano de dados (*data plane*). No nível intermediário encontra-se o plano de controle (*control plane*). E, no nível mais alto, tem-se o plano de aplicação (*application plane*). A comunicação entre os controladores e o plano de dados é mantida por meio da Interface Sul (*Southbound Interface - SBI*), localizada nos *switches SDN*, enquanto a comunicação entre as aplicações e os controladores é mantida pela Interface Norte (*Northbound Interface - NBI*), localizada no plano de controle [115]. A Figura 9 ilustra a organização dessa arquitetura.



Fonte: Adaptada de [115].

Figura 9 – Arquitetura *SDN*.

3.4.2 Plano de controle

O plano de controle ou *control plane* está localizado no centro da arquitetura da rede *SDN*, atuando de maneira intermediária entre os planos de aplicações e de dados. No plano de controle, por meio de um controlador, toda lógica e inteligência da rede é gerenciada de maneira centralizada. O controlador *SDN* é o cérebro do plano de controle, ele é um abstração de software que controla toda a rede. O controlador se comunica

com os dispositivos de rede no plano de dados usando protocolo o OpenFlow. Ele coleta informações sobre o estado da rede, como topologia, tráfego e recursos disponíveis, e toma decisões inteligentes com base nessas informações [116].

A unidade lógica de controle do tráfego de rede é por meio de fluxos *IP* no qual cada um deles possuem um conjunto de características compartilhadas, como endereços *IP* de origem e destino, portas de origem e destino, bem como protocolos de transporte. O controlador *SDN* então formula e aplica regras específicas de encaminhamento de pacotes, cada regra consiste em critérios de correspondência (*matches*) que identificam o fluxo *IP* e ações que devem ser tomadas quando o critério é satisfeito. Essas ações incluem o direcionamento do pacote para uma determinada porta de saída, a modificação de cabeçalhos ou a aplicação de políticas de serviço.

No plano de controle, o controlador *SDN* também é responsável pela implementação de políticas de rede e serviços avançados. Isso inclui a aplicação de políticas de segurança, como controle de acesso baseado em identidade, *firewall* e prevenção de ataques *DDoS* [116]. Além disso, o controlador pode oferecer serviços como balanceamento de carga, gerenciamento de largura de banda e otimização de tráfego, conforme necessário para atender às demandas específicas das aplicações e dos usuários finais .

Em suma, o plano de controle é um elemento altamente programável e adaptável em uma arquitetura *SDN*, proporcionando maior flexibilidade, escalabilidade e eficiência na operação das redes. Suas capacidades permitem a rápida adaptação a mudanças, uma gestão centralizada e a implementação de serviços personalizados, tornando-se uma peça essencial para o avanço e evolução da tecnologia de redes de próxima geração.

3.4.3 Plano de dados

O plano de dados, também denominado *data plane* ou *forwarding plane*, é uma camada crítica em redes *SDN*, responsável pela realização física do encaminhamento de pacotes de dados através da rede. Sua operação é inteiramente baseada nas regras previamente estabelecidas pelo plano de controle. O plano de dados é composto por dispositivos de encaminhamento, tais como *switches* e roteadores, que assumem a função primária de encaminhar pacotes de forma eficiente ao longo da rede. Em ambientes *SDN*, esses dispositivos são denominados *switches SDN*. Eles possuem uma interface de comunicação aberta com o controlador *SDN*, permitindo que este último envie instruções e diretrizes sobre como o tráfego deve ser tratado [117].

Os *switches SDN* contam com tabelas de fluxo, nas quais as regras de encaminhamento são armazenadas. Cada regra é constituída por critérios de correspondência (*matches*) e ações a serem executadas quando um pacote coincide com esses critérios. Uma tabela de fluxo é composta pelos campos *Match*, *Priority*, *Action*, *Counters*, *Timeout* e *Cookie*. Cada uma desses campos auxiliam no controle e gerenciamento dos fluxos. O campo

Match é utilizado para encontrar a correspondência dos pacotes que pertencem ao mesmo fluxo. A prioridade de entrada dos fluxos é garantida aplicado o campo *Priority*. Por outro lado, a regra de como os pacotes serão processados e para onde serão encaminhados são definidas no campo *Action*. O campo *Counters* mantém a quantidade de bits pacotes de um fluxo. O campo *Timeout* controla o tempo que um fluxo fica inativo e o tempo máximo que ele será mantido no *switch* até a sua exclusão. O *Cookie* é um campo escolhido pelo controlador para filtrar estatísticas, modificação ou exclusão de um fluxo.

O controlador *SDN* é o responsável pela configuração e atualização dessas tabelas de fluxo. Quando um pacote ingressa em um *switch*, ele é processado pelas tabelas de fluxo em busca de uma correspondência apropriada. Se uma correspondência for encontrada, as ações associadas são tomadas, tais como o encaminhamento do pacote para uma porta de saída específica. A comunicação entre o plano de controle e o plano de dados é realizada por meio de um protocolo de comunicação, como o OpenFlow [118]. O OpenFlow é um protocolo padronizado e aberto que possibilita ao controlador programar regras diretamente nas tabelas de fluxo dos *switches SDN*. Além disso, o protocolo permite que os *switches* comuniquem eventos e estatísticas de tráfego ao controlador, fornecendo uma visão em tempo real do estado da rede.

3.5 Considerações sobre o capítulo

Neste capítulo foram apresentados os principais conceitos utilizados como fundamentação teórica para o desenvolvimento desta tese. Inicialmente foram detalhados os conceitos relacionados a detecção de intrusão de anomalias. Na sequência, foram explorados as principais definições e conceitos relacionados as redes neurais profundas. Outro tema abordado, foram os hiperparâmetros e como eles podem afetar no treinamento de uma rede neural profunda. E por fim, foram apresentados os principais conceitos relacionados a redes definidas por software.

No próximo capítulo será apresentado o sistema para a detecção e a mitigação de anomalias em redes *SDN* aplicando redes neurais profundas, detalhando cada um dos módulos desenvolvidos, e as técnicas e soluções aplicadas.

4 Sistema proposto

Neste capítulo, é apresentado o sistema para a detecção e a mitigação de anomalias em redes *SDN* aplicando redes neurais profundas. As soluções desenvolvidas foram implementadas em uma arquitetura modular com funções bem definidas. Essa arquitetura é composta de quatro módulos: coleta de dados; processamento; caracterização e detecção; e mitigação.

A primeira solução desenvolvida, *Long Short-Term Memory* (*LSTM*) e Lógica Fuzzy para detecção e mitigação de anomalias em redes *SDN*, utiliza a *LSTM* com o objetivo de aprender e extrair padrões de curto e longo prazo para prever o comportamento normal do tráfego de rede e, em conjunto com a lógica Fuzzy, atua na detecção de ataques *DDoS* e *Portscan*.

As arquiteturas de redes neurais profundas têm um vasto poder de abstração e de extração de conhecimento de grandes volumes de dados. Apesar dessas caraterísticas, estudos indicam que as redes neurais profundas são sensíveis à detecção de ataques adversários [25, 26, 27], os quais são compostos por amostras ruidosas projetadas intencionalmente por um invasor com o objetivo de causar erros na precisão do modelo. Esses ataques podem diminuir significativamente a robustez dos modelos de aprendizagem profunda [28]. A segunda solução apresentada neste capítulo utiliza o *framework Generative Adversarial Network* (*GAN*) para a detecção de ataques *DDoS* e aplica o treinamento adversário para tornar o sistema menos sensível a ataques adversários.

As seções a seguir descrevem o funcionamento do sistema proposto, incluindo a arquitetura modular e os métodos de *DL* utilizados na detecção de eventos anômalos.

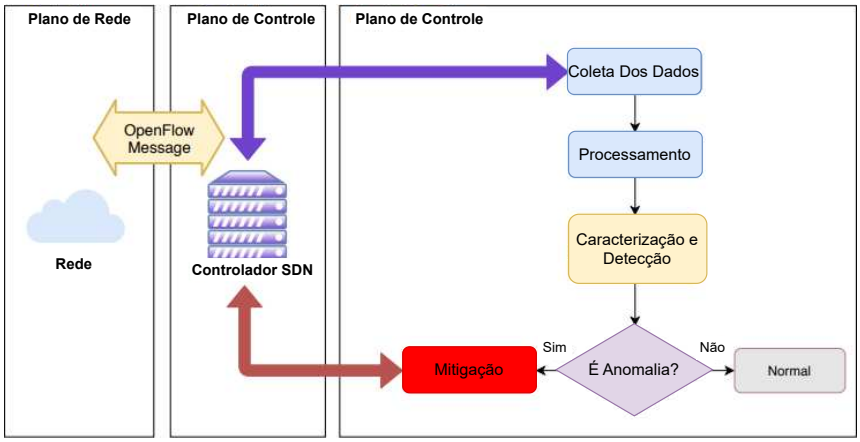
4.1 Arquitetura modular

A arquitetura modular para a implementação das soluções é composta de quatro módulos, conforme indicado a seguir:

1. Coleta dos dados: o controlador coleta fluxos *IP* das tabelas de fluxos dos *switches* a cada segundo.

2. Processamento: ocorre o agrupamento dos atributos quantitativos (*bits/s* e pacotes/s) e a transformação dos atributos de fluxos *IP* categóricos (endereços *IP* de destino/origem e números de portas) por meio do cálculo da entropia de Shannon.

3. Caracterização e detecção de anomalias: caracteriza o comportamento normal do tráfego de rede e detecta eventos anômalos com base na análise de diferentes características de fluxos *IP*.



Fonte: O próprio autor.

Figura 10 – Arquitetura da solução para a detecção e a mitigação de anomalias.

4. Mitigação: toma contramedidas para mitigar os ataques detectados e minimizar os danos provocados por eles.

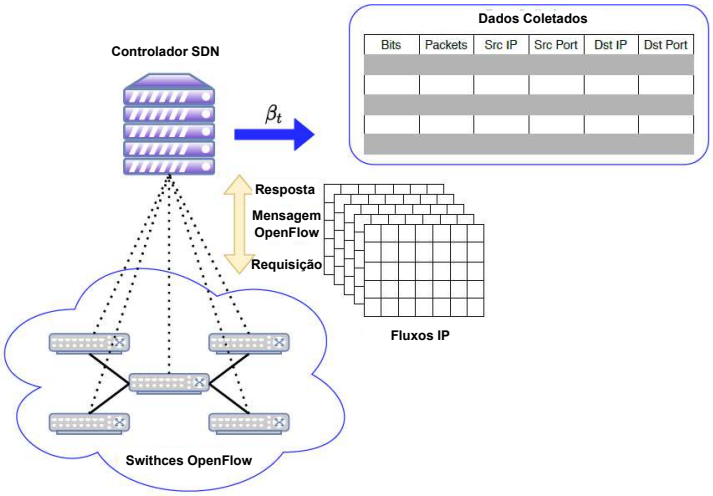
A Figura 10 ilustra o diagrama operacional da arquitetura proposta. Na sequência serão detalhados os módulos de coleta de dados, processamento e mitigação. O módulo de detecção será detalhado nas soluções desenvolvidas.

4.1.1 Coleta dos dados

A coleta dos dados é uma etapa fundamental para o monitoramento contínuo das atividades da rede. Além disso, é um pré-requisito para detectar possíveis anomalias. O sistema proposto contém um módulo para a aquisição de dados de tráfego em que as informações da rede são coletadas dos *switches* usando o protocolo OpenFlow. Esse protocolo usa o conceito baseado em fluxo para identificar o tráfego de rede, que é tratado em termos de fluxos, em vez de pacotes individuais. Conforme definido em [119], os fluxos são uma sequência de pacotes passando por um ponto de observação na rede durante um intervalo de tempo específico. Todos os pacotes em um fluxo têm propriedades comuns, como protocolo de transporte, endereços *IP* e portas de origem e de destino.

Trabalhos anteriores na literatura coletaram informações dos dados de rede usando intervalos de tempo entre 1 e 5 minutos [120, 121]. No entanto, esse intervalo de análise foi reduzido devido às altas taxas de transmissão alcançadas no cenário atual de rede. As novas soluções presentes na literatura utilizam análise de intervalos quase em tempo real [122, 123]. Dessa forma, foi reduzida a análise de intervalo de tempo para um segundo. A cada segundo, o sistema proposto solicita e analisa os registros de fluxo *IP* coletados de cada *switch*. Esse intervalo permite que o sistema reaja rapidamente contra eventos de anomalia, reduzindo o tempo de resposta.

Para todo tempo t, o controlador envia solicitações para coletar registros de fluxo *IP* para *switches* pertencentes à rede por meio de mensagens *Read-State* do OpenFlow. Após



Fonte: O próprio autor.

Figura 11 – Coleta de atributos de fluxo dos *switches* OpenFlow.

cada *switch* receber essa mensagem, é enviada uma resposta contendo cada registro de fluxo armazenado em sua tabela de encaminhamento. Esse processo é representado na Figura 11. Os dados coletados podem ser definidos como um conjunto $\beta_t = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, em que cada α_i é um registro de fluxo composto dos atributos apresentados na Tabela 2.

Tabela 2 – Atributos de fluxos coletados.

Flow Feature	**Description**
$x^{bits}_{\in\alpha_i}$	Quantidade de *bits* pertencentes ao fluxo α_i
$x^{pacotes}_{\in\alpha_i}$	Quantidade de pacotes pertencentes ao fluxo α_i
$x^{srcIP}_{\in\alpha_i}$	Endereço de *IP* de origem
$x^{srcPort}_{\in\alpha_i}$	Número da porta de origem
$x^{dstIP}_{\in\alpha_i}$	Endereço de *IP* de destino
$x^{dstPort}_{\in\alpha_i}$	Número da porta de destino

Os atributos de fluxo coletados podem ser classificados como quantitativos (*bits* e pacotes) e qualitativos (endereço *IP* de origem, endereço *IP* de destino, porta de origem e porta de destino). Os atributos quantitativos de fluxo fornecem informações sobre o volume de tráfego, o que é essencial para entender a quantidade de tráfego transportado na rede. Por outro lado, as características de fluxo qualitativo nos permitem entender quais dispositivos se comunicam e quais aplicativos estão sendo acessados.

4.1.2 Processamento

Os registros de fluxo *IP* coletados precisam ser processados para extrair características específicas que auxiliam na abordagem de detecção de anomalias. O módulo de processamento de dados é responsável por agrupar os atributos de fluxo em cada intervalo de análise. A cada segundo, os seguintes atributos são processados:

- *bits/s.*
- *pacotes/s.*
- *entropia IP de origem.*
- *entropia porta de origem.*
- *entropia IP de destino*
- *entropia porta de destino.*

Esses atributos de fluxos foram amplamente analisados e empregados em trabalhos anteriores presentes na literatura [124]. Eles têm sido utilizados na caracterização de tráfego de rede de alta velocidade e os resultados alcançados para a detecção de anomalias têm sido eficazes.

As taxas de *bits* e pacotes por segundo, atributos quantitativos, são processadas por:

$$bits/s = \sum_{j=1}^{|\beta_t|} x_j^{bits} \tag{4.1}$$

$$pacotes/s = \sum_{j=1}^{|\beta_t|} x_j^{pacotes} \tag{4.2}$$

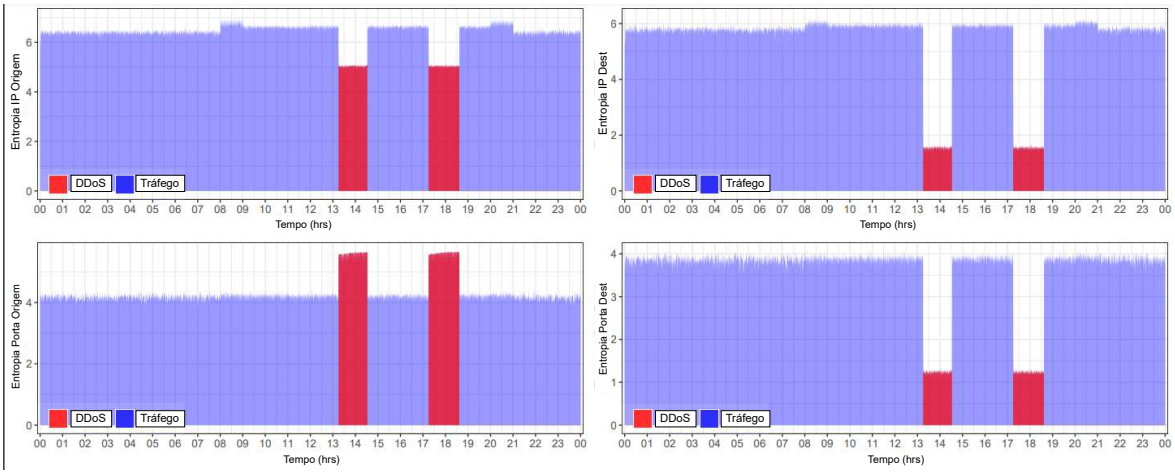
Os endereços *IP* e os números de portas são atributos categóricos. No entanto, o módulo de detecção aplica uma rede neural profunda para detectar anomalias. As redes neurais profundas exigem que as variáveis de entrada sejam numéricas. Assim, os atributos qualitativos, *IP* e portas, devem ser convertidos para o tipo numérico. Uma transformação simples é agrupar esses dados calculando a entropia. Neste módulo, aplicamos a Entropia de Shannon [125], que destaca o grau de concentração ou dispersão dos atributos durante o intervalo de tempo analisado. Dado um conjunto de um atributo qualitativo, como $x^{srcIP} = \{x_1, x_2, \ldots, x_n\}$, em que um x_i representa o número de ocorrências do endereço *IP* de origem i em determinado intervalo de tempo t. A Entropia de Shannon $H(\cdot)$ para o atributo de fluxo x^{srcIP} é definida como:

$$H(x^{srcIP}) = -\sum_{i=1}^{n} \left(\frac{x_i}{S}\right) log_2 \left(\frac{x_i}{S}\right), \tag{4.3}$$

em que $S = \sum_{i=1}^{n} x_i$ é a soma de todas as ocorrências dos elementos presentes no intervalo de tempo analisado t. O cálculo de entropia para os outros atributos qualitativos [como $H(x^{srcPort})$, $H(x^{dstIP})$ e $H(x^{dstPort})$] é realizado da mesma maneira, como apresentado na Equação (4.3).

As informações sobre o grau de concentração ou dispersão de determinado recurso de fluxo podem ajudar durante a fase de detecção de anomalias. Por exemplo, em uma ocorrência de ataque *DDoS*, o endereço *IP* de destino e a distribuição do número da porta

de destino podem ficar concentrados devido à alta quantidade de conexões solicitadas pelos atacantes. A concentração e a dispersão dos atributos de fluxo afetados durante a ocorrência de um ataque *DDoS* são ilustradas na Figura 12. Essa figura apresenta os resultados das análises de entropia *IP* de origem, entropia *IP* de destino, entropia de porta de origem e entropia de porta de destino para um dia de tráfego de rede usado para avaliar o sistema proposto. Os intervalos azuis representam a entropia medida durante o comportamento normal da rede. Como pode ser visto, o valor da entropia permanece contínuo, sem variações significativas nas faixas de normalidade. Os intervalos em vermelho, ao contrário, representam a ocorrência de ataques *DDoS*. Há uma concentração nesses intervalos dos endereços *IP* de origem e de destino e da porta de destino, conforme mostrado. Por outro lado, há uma dispersão da entropia da porta de origem, porque vários atacantes usam portas de origem aleatórias.



Fonte: O próprio autor.

Figura 12 – Análise dos atributos de entropia durante ataques de *DDoS*.

4.1.3 Mitigação

A detecção e a identificação de anomalias são etapas essenciais para garantir a operabilidade e os serviços disponíveis pelos sistemas de rede. Após a ocorrência de um evento anômalo, deve ser adotado um mecanismo para minimizar os efeitos provocados por ele. O processo usual para diminuir os efeitos causados por ataques ocorre por meio da mitigação, aplicando políticas autonômicas sem a necessidade da interferência humana. Seu propósito é garantir a resiliência e a operabilidade da rede. Desse modo, o sistema proposto é composto de um módulo responsável por identificar os fluxos anômalos e aplicar políticas de mitigação.

As políticas de mitigação são estruturadas utilizando o modelo **Evento-Condição-Ação** (**ECA**), que é considerado adequado para o gerenciamento dinâmico de políticas. Nesse modelo, o **Evento** refere-se a uma anomalia específica e está associado a um con-

junto de regras, que são descritas como um conjunto de **Condição** que corresponde ao contexto em que a anomalia ocorreu. Por fim, a **Ação** é a contramedida tomada com relação aos fluxos identificados como maliciosos

O método principal utilizado nas aplicações para a mitigação de ataques em ambientes *SDN* é modificar a entrada da tabela de fluxo ou adicionar uma nova entrada na tabela do *switch*. Após a detecção de um ataque, algumas características devem ser identificadas, por exemplo, endereço *IP* de origem, endereço *IP* de destino, número da porta de origem, número da porta de destino e o tipo de protocolo. Essas características auxiliam na identificação do atacante e são fundamentais para tomar as contramedidas apropriadas para minimizar o dano de um ataque. Uma nova entrada na tabela de fluxos pode ser instalada com base em uma ou mais dessas características, sinalizando que os pacotes que pertencem ao fluxo são de um ataque. As ações tomadas podem ser o descarte desses pacotes, o bloqueio do tráfego anômalo e/ou um redirecionamento a um *honeypot* [126].

Fundamentado nos conceitos apresentados, o módulo de mitigação do sistema é composto de duas políticas para mitigar as anomalias detectadas. Após o módulo de detecção disparar um alarme, o módulo de mitigação entra em ação. O primeiro passo é identificar os fluxos suspeitos do intervalo de análise. Essa identificação é feita com base na análise dos endereços *IP* e das portas que compõem o intervalo anômalo. São considerados fluxos suspeitos todos aqueles que se destinam ao endereço de *IP* que mais recebeu fluxos.

Algorithm 1 Processo de Mitigação.

Require: Fluxos suspeitos
Ensure: Descarte de pacotes anômalos

```
1: if Ataque de DDoS then
2:     Identificar o endereço de IP de destino que mais recebeu fluxos
3:     Identificar nesses fluxos os endereços IP dos atacantes que têm a mesma porta de
   destino
4:     if IPs e Portas estão na Safe List then
5:         Encaminhar os pacotes
6:     else
7:         Descartar pacotes
8: if Ataque Portscan then
9:     Identificar o endereço de IP de destino que mais recebeu fluxos
10:    Identificar nesses fluxos o IP de origem que apresenta maior variedade de porta
   de destino
11:    if IPs e Portas estão na Safe List then
12:        Encaminhar os pacotes
13:    else
14:        Descartar pacotes
```

Com a identificação dos fluxos suspeitos, caso o **Evento** disparado pelo módulo de detecção for um ataque de *DDoS*, será feito um descarte dos fluxos com base nos ende-

reços *IP* de origem que aparecem com maior frequência nos fluxos suspeitos e que têm, simultaneamente, a mesma porta de destino. Quando o **Evento** disparado for um ataque de *Portscan*, o processo de identificação do atacante é feito por meio do endereço de *IP* de origem que apresenta a maior variedade de portas de destino. Esse *IP* é considerado atacante e todos os seus fluxos serão descartados. O processo de mitigação é mostrado no Algoritmo 1.

4.2 Sistema 1: *Long Short-Term Memory* e Lógica Fuzzy para detecção e mitigação de anomalias em redes *SDN*

O sistema proposto nesta seção tem como finalidade a caracterização do tráfego e a detecção e a mitigação de ataques de *DDoS* e *Portscan* em Redes Definidas por *Software*. O sistema utiliza como princípio o conceito de *Digital Signature of Network Segments* (*DSNS*), introduzido por Proença *et al.* [120, 127]. O modelo de assinaturas digitais proposto pelos autores utilizava uma técnica estatística com o objetivo de criar uma caracterização do segmento de rede analisado. A assinatura gerada foi utilizada como base comparativa em um módulo de detecção de anomalias que a comparava com o tráfego real de rede. A caracterização proposta por Proença *et al.* foi idealizada para o ambiente tradicional de rede e utilizava dados históricos de várias semanas anteriores, coletados por meio do protocolo *SNMP* (*Simple Network Management Protocol*). Por outro lado, a caracterização proposta nesta tese utiliza atributos de fluxos *IP* coletados dos *switches*, e a predição da assinatura de rede é realizada empregando uma janela deslizante com intervalos de um segundo. Consequentemente, o sistema proposto dispensa a utilização de uma base contendo as assinaturas das anomalias. Por meio do perfil normal da rede, é possível reconhecer comportamentos que diferem do esperado e auxiliar na fase de detecção de anomalias da metodologia apresentada neste trabalho.

O processo de caracterização e detecção de anomalias é dividido em duas etapas:

1. Previsão do comportamento normal do tráfego de rede, utilizando a arquitetura de rede neural profunda *LSTM* e a definição dos limiares de normalidade.

2. Aplicação da lógica Fuzzy para determinar se existem anomalias, utilizando como parâmetro o tráfego previsto e os limiares definidos na fase anterior.

4.2.1 Etapa 1: *Long Short-Term Memory* para caracterização do tráfego de rede

A previsão do tráfego tem como finalidade gerar a assinatura do comportamento normal e, assim, seu emprego é essencial para o gerenciamento e a segurança da rede. A caracterização torna mais confiáveis e seguras as decisões de gerência com relação a possíveis problemas que venham a ocorrer na rede. Para esse propósito, na fase de caracterização, é utilizada a arquitetura de rede neural recorrente profunda *Long Short-Term Memory* (*LSTM*). Introduzida por Hochreiter e Schmidhuber [89], a *LSTM* é uma arquitetura especial de rede recorrente, com capacidade de aprender dependências de longo prazo. A estrutura de uma célula *LSTM* é ilustrada na Figura 13.

Conforme pode ser observado, a cada instante de tempo t, a célula é controlada por diversos *gates* que podem tanto manter quanto "resetar" o valor de acordo com o estado do *gate*. Na célula, três *gates* são aplicados, o *forget gate* ($\mathbf{f}_t$), o *input gate* ($\mathbf{i}_t$) e o *output gate* ($\mathbf{o}_t$). Além disso, há um *gate* de modulação de entrada, chamado *candidate value* ($\mathbf{c'}_t$). Os *gates* podem ser descritos da seguinte forma:

$$\mathbf{i}_t = \sigma(\mathbf{W}_i\mathbf{a}_t + \mathbf{U}_i\mathbf{h}_{t-1} + b_i) \tag{4.4}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f\mathbf{a}_t + \mathbf{U}_f\mathbf{h}_{t-1} + b_f) \tag{4.5}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o\mathbf{a}_t + \mathbf{U}_o\mathbf{h}_{t-1} + b_o) \tag{4.6}$$

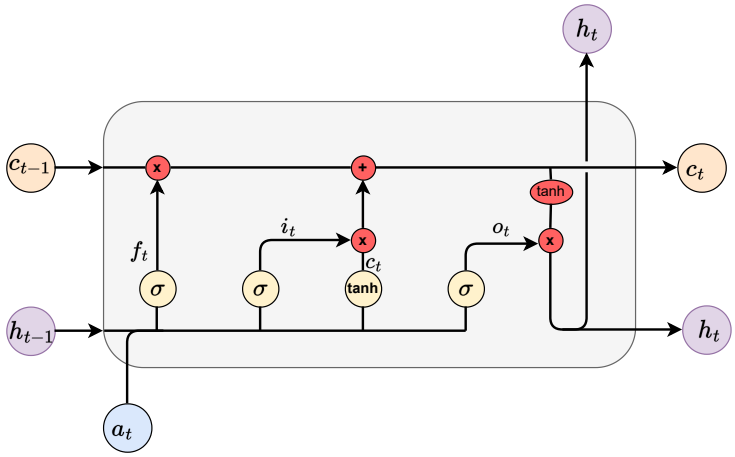
$$\mathbf{c'}_t = tanh(\mathbf{W}_{c'}\mathbf{a}_t + \mathbf{U}_c\mathbf{h}_{t-1} + b_{c'}) \tag{4.7}$$

em que $\mathbf{W} \in \mathbb{R}^{u\times n}$, $\mathbf{U} \in \mathbb{R}^{u\times u}$, $b \in \mathbb{R}^u$ são as matrizes de pesos sinápticos e os vetores de *bias*. Os índices u e n referem-se ao número de unidades ocultas e ao tamanho do vetor de entrada, respectivamente. A entrada atual é representada por $\mathbf{a}_t$; $\mathbf{c'}_t$ é um vetor com novos candidatos a serem adicionados ao estado atual da célula; $\mathbf{h}_{t-1}$ é a saída anterior da *LSTM* no instante de tempo $t-1$; $\sigma(\cdot)$ e $\tanh(\cdot)$ são as respectivas funções de ativação, Sigmoide e Tangente Hiperbólica. O primeiro passo na *LSTM* é decidir quanto do valor de memória anterior será removido do estado da célula. Essa decisão é tomada pelo *forget gate*. O próximo passo é determinar quanto da nova informação será armazenada, que é realizado pelo *input gate*. Em seguida, o estado da célula é atualizado e definido pela seguinte expressão:

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{c'}_t \tag{4.8}$$

em que $\odot$ denota o produto Hadamard. E a saída $\mathbf{h}_t$ da *LSTM* é definida por:

$$\mathbf{h}_t = \mathbf{o}_t \odot tanh(\mathbf{c}_t) \tag{4.9}$$



Fonte: O próprio autor.

Figura 13 – Organização estrutural de uma célula *LSTM*.

4.2.1.1 Previsão do tráfego de rede

Obter uma previsão próxima ao comportamento real é um passo importante para a detecção de tráfego anômalo, pois a assinatura gerada delimita os limites de normalidade de uma amostra do tráfego em determinado instante do segmento de rede observado [128].

As caracterizações das assinaturas são geradas a partir de dados de fluxos *IPs* que são coletados dos *switches* da rede *SDN* pelo controlador, utilizando o protocolo OpenFlow. Entre os atributos coletados pelo controlador, foram selecionados os seguintes: *bits/s*, pacotes/s, endereço *IP* de origem, endereço *IP* de destino, porta de origem e porta de destino. Esses atributos de fluxos foram analisados e empregados em trabalhos anteriores na caracterização do tráfego de redes de altas velocidades e apresentaram bons resultados para descrever e compreender melhor o comportamento da rede [129, 130]. As dimensões *bits* e pacotes são atributos quantitativos, ou seja, de volume, que são capazes de fornecer informações relativas à quantidade de dados que estão trafegando ao longo da rede. Os demais são atributos nominais e fornecem informações qualitativas, isto é, permitem compreender quais dispositivos estão se comunicando e quais serviços estão sendo acessados por eles. A utilização desses atributos é fundamental para identificar possíveis atacantes

e é indispensável na utilização do módulo de mitigação para minimizar os danos causados por um ataque.

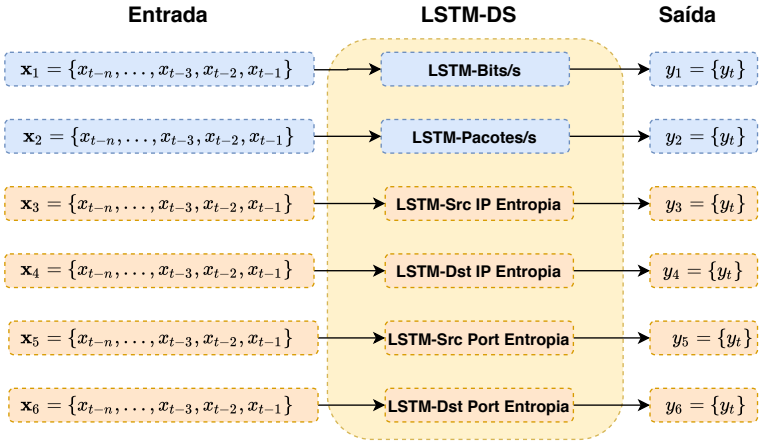
Os atributos *IP* e porta são dados nominais e, para realizar uma análise quantitativa é necessária a sua transformação por meio do cálculo da entropia. Portanto, neste trabalho foi empregada a Entropia de Shannon [125], que permite extrair informações da concentração e da dispersão desses atributos de fluxo, conforme descritos na seção anterior.

Posteriormente, ao garantir que todos os atributos de fluxos coletados estão representados de maneira quantitativa, inicia-se o processo de geração das assinaturas do tráfego. O problema de previsão do tráfego de rede, utilizando a *LSTM*, pode ser definido do seguinte modo. Considere, no instante t, o conjunto de dados $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_d)$, em que cada $\mathbf{x}_i$ é um vetor de atributo de fluxo definido como:

- $\mathbf{x}_1$: bits/s;
- $\mathbf{x}_2$: pacotes/s;
- $\mathbf{x}_3$: entropia *IP* de origem;
- $\mathbf{x}_4$: entropia *IP* de destino;
- $\mathbf{x}_5$: entropia porta de origem;
- $\mathbf{x}_6$: entropia porta de destino.

A *LSTM* foi aplicada para modelar um problema de previsão de série temporal univariada. Foi aplicada uma *LSTM* para cada um dos atributos de fluxos definidos anteriormente, ou seja, cada *LSTM* será responsável por prever a assinatura do comportamento normal de cada atributo $\mathbf{x}_i$. O modelo *LSTM* aprenderá uma função que mapeia uma sequência de n observações passadas como entrada para uma observação de saída [131]. Por exemplo, no instante t, dada uma sequência de entrada de n observações passadas que consiste no vetor de bits $\mathbf{x}_1 = \{x_{t-n}, \ldots, x_{t-3}, x_{t-2}, x_{t-1}\}$, produz-se uma saída $y_1 = \{y_t\}$ que representa a previsão do comportamento para o atributo de fluxo de *bits*. A Figura 14 ilustra o modelo *LSTM For Digital Signature* (*LSTM-DS*), proposto neste trabalho.

As redes *LSTM* são projetadas para lidar com dados sequenciais, devido à sua capacidade de manter a memória de longo prazo. Nos últimos anos, a *LSTM* tem sido amplamente utilizada na previsão de séries temporais e provou ser superior aos algoritmos matemáticos tradicionais [132, 133, 134]. Além disso, a *LSTM* é uma técnica poderosa que pode representar a relação entre eventos atuais e anteriores, além de melhorar a previsão do tráfego de rede.



Fonte: O próprio autor.

Figura 14 – Módulo de caracterização do tráfego de rede utilizando seis redes *LSTM*.

Apesar de utilizar seis redes *LSTM*, uma para cada dimensão de fluxo, o processo de treinamento das redes é uma tarefa *off-line*, o custo computacional para o treinamento é alto, porém, não é crítico para a sua aplicação [135]. Portanto, na fase de operação, com as redes *LSTM* treinadas, o processo de previsão do tráfego é imediato. O Algoritmo 2 ilustra o processo de operação do módulo *LSTM-DS*.

Durante a fase de treinamento de uma estrutura de rede *LSTM*, é necessário definir o número de unidades ocultas (neurônios) e a quantidade de amostras de observações passadas, o *time step*, para prever o comportamento de um dado instante de análise do tráfego. A escolha desses hiperparâmetros é detalhada na subseção (5.2.2).

Algorithm 2 LSTM-DS: Fase de operação

Require: $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_d)$
Ensure: $\mathbf{y} = (y_1, y_2, \ldots, y_d)$

1: $y_1 = \text{LSTM-bits}(\mathbf{x}_1)$
2: $y_2 = \text{LSTM-Pacotes}(\mathbf{x}_2)$
3: $y_3 = \text{LSTM-SrcIPEntropia}(\mathbf{x}_3)$
4: $y_4 = \text{LSTM-DstIPEntropia}(\mathbf{x}_4)$
5: $y_5 = \text{LSTM-SrcPortEntropia}(\mathbf{x}_5)$
6: $y_6 = \text{LSTM-DstPortEntropia}(\mathbf{x}_6)$

7: $\mathbf{y} = (y_1, y_2, y_3, y_4, y_5, y_6)$
8: **return y**

O tráfego previsto não será exatamente igual ao tráfego real, portanto é necessário determinar limiares de normalidade entre o tráfego previsto e o tráfego real. A desigualdade de Bienaymé-Chebyshev é utilizada para fazer essa definição, uma vez que determina um limiar da porcentagem de dados que existem dentro do número k de desvios padrão com relação à média. A desigualdade pode ser aplicada para a detecção de *outliers* [136] quando não se conhece a distribuição dos dados.

A fórmula para a desigualdade de Bienaymé-Chebyshev é representada na Equação (4.10):

$$P(\mid X - \mu \mid \geq k\theta) \leq \frac{1}{k^2}, \tag{4.10}$$

em que X é uma variável aleatória, μ é a média, $k > 0$ é o parâmetro de desvios e θ é o desvio padrão. Definindo o parâmetro $k = 4{,}47$ na Equação (4.10), a probabilidade resultante será igual a 0,05, que é o ponto de corte usual de significância estatística para verificar a discrepância de uma hipótese em relação aos dados observados [137].

4.2.2 Fase 2: Lógica Fuzzy para detecção de anomalias

A fase de detecção de anomalias é responsável por detectar e identificar atividades causadas por agentes maliciosos que possam provocar comportamentos anômalos. Nessa fase, é necessário utilizar técnicas capazes de diferenciar o processo normal de operação do tráfego de possíveis ataques que venham a ocorrer contra a rede.

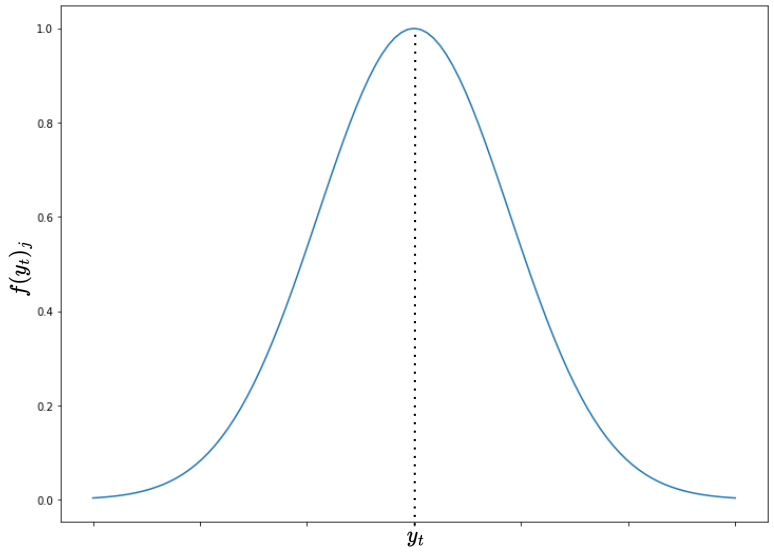
De acordo com Wu e Banzhaf [138], a lógica Fuzzy é apropriada para detecção de anomalias em redes por duas principais razões: primeiro, o problema de detecção de anomalias envolve inúmeros atributos numéricos, que são coletados e derivados estatisticamente, o que pode gerar erros durante o processo de detecção; segundo, os modelos que geram um perfil normal do comportamento da rede precisam determinar limiares entre o comportamento normal e o anômalo. No entanto, esse intervalo não é bem definido e mudanças pequenas no comportamento podem causar falsos alarmes. Considerando esses fatores, a lógica Fuzzy foi empregada nessa fase para auxiliar na tomada de decisão na detecção de anomalias.

Na lógica clássica, uma única proposição pode assumir valores como verdadeiro ou falso. Por outro lado, a teoria dos conjuntos Fuzzy introduziu uma nova concepção: as proposições podem assumir valores entre zero e um. Esse conceito é denominado grau de pertinência. Introduzida por Zadeh em 1965 [139], a teoria dos conjuntos Fuzzy fornece uma ferramenta matemática capaz de auxiliar na tomada de decisões em um ambiente contendo variáveis com imprecisão, incerteza e informação incompleta.

Um conjunto Fuzzy pode ser definido como (S,f), em que S é um conjunto qualquer e f representa a função de pertinência. Para cada elemento x pertencente a S, o valor $f(x)$ define o grau de pertinência de x no conjunto (S,f). O elemento x é considerado não incluso se $f(x) = 0$, totalmente incluso se $f(x) = 1$ e membro difuso se $0 < f(x) < 1$. Um exemplo de função de pertinência é a Gaussiana, que é definida por:

$$f(x) = e^{\frac{-(x-m)^2}{2\phi^2}} \tag{4.11}$$

em que m é a média e ϕ é o desvio padrão do conjunto S. Essa função de pertinência fornece resultados suaves e diferentes de zero em todos os pontos de definição de seu domínio.



Fonte: O próprio autor.

Figura 15 – Função de pertinência Gaussiana.

4.2.2.1 Detecção de anomalias

O modelo proposto para a detecção de anomalias neste trabalho utiliza o tráfego passado, as assinaturas previstas pela *LSTM-DS* e a lógica Fuzzy. O primeiro passo é a "fuzzyficação" das entradas para cada um dos seis atributos de fluxos em análise, aplicando a função de pertinência, que, neste trabalho, é uma modificação da função de pertinência Gaussiana, definida por:

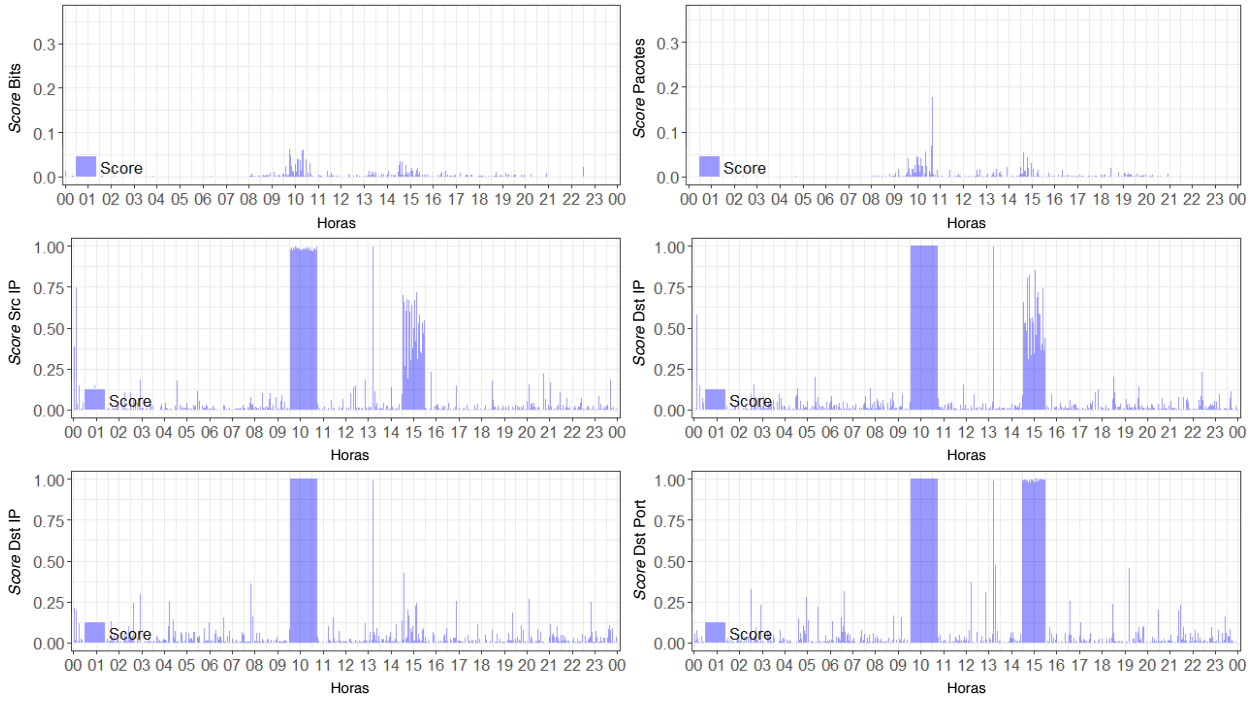
$$f(y_t)_j = e^{\frac{-(x_t - y_t)^2}{2\hat{\sigma}_t^2}} \tag{4.12}$$

em que x_t é o tráfego real, y_t é a assinatura prevista pela *LSTM* e $\hat{\sigma}_t$ é o limiar gerado pela desigualdade de Bienaymé-Chebyshev do atributo de fluxo j no instante de tempo t. Quanto mais próximo o tráfego real estiver da assinatura prevista, maior será o grau de pertinência de x_t estar contido no conjunto do tráfego normal do atributo de fluxo j. A função de pertinência $f(y_t)_j$ está ilustrada graficamente na Figura 15.

A Equação 4.12 determina o grau de pertinência do conjunto do tráfego normal. No entanto, para detecção de uma anomalia devemos, aplicar o seu complemento, definido por:

$$f'_j = 1 - f_j \tag{4.13}$$

O resultado de f'_j representa o *score* de anomalia do atributo de fluxo j. Os *scores* de anomalias são utilizados para classificar o comportamento do tráfego para determinado instante de análise. O processo de "desfuzzyficação" determina se o tráfego é "normal", "*Portscan*" ou "*DDoS*", definidos pelas seguintes regras:



Fonte: O próprio autor.

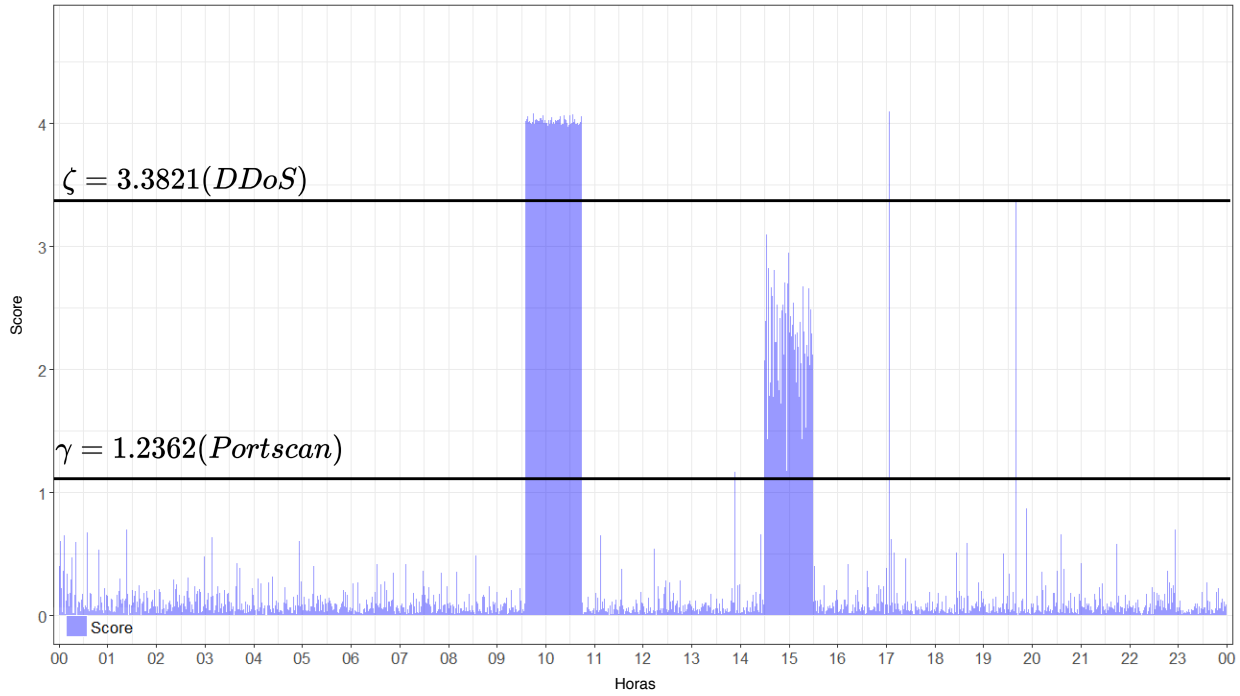
Figura 16 – *Score* de anomalia por atributo de fluxo.

$$\textbf{Rule 1:} \quad IF \quad \sum_{j=1}^{6} f'_j < \gamma \quad THEN \quad \text{“}normal\text{”} \tag{4.14}$$

$$\textbf{Rule 2:} \quad IF \quad \sum_{j=1}^{6} f'_j \geq \gamma \quad AND \sum_{j=1}^{6} f'_j < \zeta \quad THEN \quad \text{“}Portscan\text{”} \tag{4.15}$$

$$\textbf{Rule 3:} \quad IF \quad \sum_{j=1}^{6} f'_j \geq \zeta \quad THEN \quad \text{“}DDoS\text{”} \tag{4.16}$$

Os valores dos limiares para γ e ζ foram definidos como 1.2362 e 3.3821, respectivamente. Esses valores foram avaliados por meio da acurácia e são detalhados na Seção 5.2.2. A Figura 16 ilustra o processo de “fuzzyficação” para cada um dos atributos de fluxos, obtendo por meio da função definida na Equação (4.13), os *scores* de anomalias dos respectivos atributos. Por outro lado, a Figura 17 representa o processo de “desfuzzyficação”, aplicando as regras de inferências definidas anteriormente. Pode-se notar que, durante a ocorrência dos ataques, os valores obtidos do somatório dos *scores* são elevados, permitindo a aplicação das regras de inferência para a detecção dos eventos anômalos.



Fonte: O próprio Autor.

Figura 17 – Somatório dos *scores* de anomalia dos seis atributos de fluxo.

4.2.3 Análise de complexidade computacional da *LSTM*

A análise de complexidade de uma rede neural na sua fase de operação - após o seu treinamento, em que são encontrados os pesos sinápticos - é basicamente verificar qual é o custo de propagar o sinal na camada de entrada até a camada de saída. Isso também se aplica ao caso de calcular a complexidade computacional (CC) de uma rede *LSTM*.

Na *LSTM*, o custo computacional é analisado considerando cada um dos *gates* presentes em sua arquitetura. Considerando os *gates* definidos na subseção 4.2.1, tem-se as seguintes equações:

$$\mathbf{i}_t = \sigma(\mathbf{W}_i\mathbf{a}_t + \mathbf{U}_i\mathbf{h}_{t-1} + b_i) \tag{4.17}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f\mathbf{a}_t + \mathbf{U}_f\mathbf{h}_{t-1} + b_f) \tag{4.18}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o\mathbf{a}_t + \mathbf{U}_o\mathbf{h}_{t-1} + b_o) \tag{4.19}$$

$$\mathbf{c'}_t = tanh(\mathbf{W}_{c'}\mathbf{a}_t + \mathbf{U}_c\mathbf{h}_{t-1} + b_{c'}) \tag{4.20}$$

O primeiro passo é analisar cada uma das operações presentes nas equações definidas, observando a dimensionalidade e os tipos de dado (matrizes e vetores). Iniciando pelo cálculo de $\mathbf{i}_t$ (*input gate*). O cálculo em termos de complexidade é o mesmo para os *gates* $\mathbf{f}_t$ (*forget gate*), $\mathbf{o}_t$ (*output gate*) e $\mathbf{c'}_t$(*candidate value*).

$$\mathbf{i}_t = \sigma(\mathbf{W}_i\mathbf{a}_t + \mathbf{U}_i\mathbf{h}_{t-1} + b_i)$$

- $\mathbf{W}_i \in \mathbb{R}^{u\times n}$ é a matriz que representa os pesos sinápticos do *input gate*;
- $\mathbf{a}_t \in \mathbb{R}^n$ é o vetor de entrada de tamanho n;
- $\mathbf{U} \in \mathbb{R}^{u\times u}$ é a matriz para as conexões recorrentes;
- $\mathbf{h}_{t-1} \in \mathbb{R}^u$ é a saída do instante de tempo t-1;
- $b_i \in \mathbb{R}^u$ vetor de *bias*.

Estimando o custo de cada operação, temos:

- $\mathbf{W}_i\mathbf{a}_t$ tem complexidade de tempo $\mathcal{O}(u \times n)$ (multiplicação matriz-vetor);
- $\mathbf{U}_i\mathbf{h}_{t-1}$ tem complexidade de tempo $\mathcal{O}(u^2)$ (multiplicação matriz-matriz de dimensão u)
- mais o custo $\mathcal{O}(u)$ para somar o bias b_i.

Desse modo, a complexidade de $\mathbf{i}_t$ é dada por:

$$\mathcal{O}(u \times n + u^2 + u) = \mathcal{O}(u(n + u + 1)) \tag{4.21}$$

Avaliando a complexidade de tempo de $\sigma()$, a função de ativação Sigmoide é obtida aplicando a seguinte operação:

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Assumindo que $\frac{1}{1+e^{-z}}$ tem complexidade de tempo linear, obtemos a complexidade $\mathcal{O}(u)$.

A mesma complexidade de tempo se aplica aos *gates* $\mathbf{f}_t$ (*forget gate*), $\mathbf{o}_t$ (*output gate*) e $\mathbf{c'}_t$(*candidate value*), então é multiplicado o resultado encontrado em 4.21 por 4. A complexidade de tempo para os *gates* pode ser dada por:

$$\mathcal{O}(4u(n + u + 1)) \tag{4.22}$$

Além disso, é considerada a complexidade de tempo para atualizar o estado da célula, dada pela operação:

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{c'}_t \tag{4.23}$$

em que $\mathbf{c}_t \in \mathbb{R}^u$ e $\odot$ denota produto Hadamard, logo a operação tem complexidade de tempo $\mathcal{O}(2 \times u)$.

A saída da rede $\mathbf{h}_t$ é definida por:

$$\mathbf{h}_t = \mathbf{o}_t \odot tanh(\mathbf{c}_t) \tag{4.24}$$

e tem complexidade de tempo $\mathcal{O}(2 \times u)$.

Portanto, a complexidade de tempo total da *LSTM* na sua fase de operação é:

$$\mathcal{O}(4u(n + u + 1) + 4u) \tag{4.25}$$

4.3 Sistema 2: *Adversarial Deep Learning* para detecção e defesa contra ataques *DDoS* em ambientes *SDN*

Apesar das inúmeras aplicações que utilizam métodos de *Deep Learning* em *NIDS*, estudos recentes demonstraram que redes neurais profundas são sensíveis a exemplos adversários (*adversarial examples*) executados por agentes maliciosos para fazer com que os métodos de *DL* cometam erros na detecção de ataques [25, 26, 27]. Exemplos adversários são amostras com ruídos projetadas intencionalmente por um atacante para que uma rede neural profunda a classifique incorretamente, ou seja, classificar um ataque como uma amostra normal. Os exemplos adversários tornaram-se uma vulnerabilidade nos sistemas que empregam as arquiteturas de redes neurais profundas, principalmente nos sistemas de segurança da informação [28, 31].

Em [29], os autores investigam exemplos adversários e sugerem que essa vulnerabilidade pode ser minimizada por meio do emprego do treinamento adversário (*adversarial training*). Em termos práticos, isso significa adicionar exemplos de dados adversários ao conjunto de dados original e usá-lo na fase de treinamento do modelo. Uma técnica que têm sido aplicada para esse propósito é o *framewok Generative Adversarial Network* (GAN), que permite gerar exemplos adversários empregando *adversarial training*. Ao treinar o gerador e o discriminador simultaneamente de forma concorrente, o discriminador melhora as taxas de detecção usando os exemplos adversários gerados e se atualiza contra eles [32].

De acordo com estudos anteriores [140, 114], os ataques *DDoS* são comumente realizados por agentes maliciosos contra sistemas de rede. A centralização do controle da rede em um controlador na arquitetura *SDN* torna-se uma vulnerabilidade explorada por ataques de *DDoS*. Dessa forma, é apresentado um sistema de detecção de anomalias baseado no treinamento de adversário, aplicando o *framework* (*GAN*) para detecção e defesa contra ataques *DDoS* recentes. Nesse sentido, foi proposto um sistema que atua no plano de aplicação para detectar e mitigar essa ameaça.

4.3.1 *Adversarial Deep Learning Anomaly Detection*

O método proposto, *Adversarial Deep Learning Anomaly Detection*, busca detectar e identificar as atividades de agentes maliciosos que podem causar comportamentos anô-

malos. Para esse módulo, é necessário utilizar técnicas capazes de diferenciar a operação normal de possíveis ataques contra a rede. As redes neurais profundas (*DNN*) têm sido aplicadas em sistemas de detecção de anomalias [141, 74]. Esses sistemas obtiveram desempenho superiores a outros sistemas clássicos de aprendizagem de máquina, embora as abordagens *DNN* sofram com exemplos adversários. Nesse módulo, foi aplicado o treinamento adversário, utilizando o *framework GAN*.

O *framework GAN* proposto por Goodfellow *et al.* [32] treina simultaneamente dois modelos por meio de um processo adversário. Ele é estruturado por dois modelos de redes neurais: um modelo generativo (G) e um discriminativo (D). Ambos os modelos competem entre si de forma antagônica. Considerando isso, tal competição consiste em um jogo *minimax* de dois jogadores de acordo com o cenário da teoria dos jogos. O modelo G gera amostras falsas de uma distribuição de ruído semelhante ao conjunto de dados original. Por outro lado, o modelo D determina a probabilidade de que as amostras pertençam ao conjunto de dados original.

O modelo G constrói um mapeamento de um ruído p_z para um espaço de dados $G(z)$ e aprende uma distribuição generativa p_g sobre os dados x. Para aumentar a taxa de erro de D, as amostras falsas devem ser o mais semelhantes possível à distribuição real p_{data}. A Equação (4.26) representa a função objetivo do modelo G:

$$min\frac{1}{2}\,\mathbb{E}_{z\sim p_z}\,log(1-D(G(z))) \tag{4.26}$$

onde $z\sim p_z$ são dados da distribuição gerada por G e $D(G(z))$ indica a probabilidade de D determinar os dados gerados por G. O modelo G atinge seu treinamento minimizando a Equação (4.26), o que significa que o modelo D prevê os dados gerados $G(z)$ com alta probabilidade.

O modelo D visa classificar se uma amostra é um dado real ou um dado gerado. Assim, a função objetivo de D é definida na Equação (4.27):

$$Max\frac{1}{2}\,\mathbb{E}_{x\sim pdata}\,logD(x)+\frac{1}{2}\,\mathbb{E}_{z}\,log(1-D(G(z))) \tag{4.27}$$

onde $D(x)$ determina a probabilidade dos dados reais e $x\sim p_{data}$ são os dados da distribuição original. A eficácia do modelo D é alcançada maximizando a Equação (4.27).

Assim, considerando o conflito entre os dois modelos, o *framework GAN* pode ser definido como um jogo *minimax*. Ambos os modelos melhoram continuamente sua eficácia até que um equilíbrio seja alcançado. Na Equação (4.28), é formalizado o jogo *minimax*:

$$minMax\frac{1}{2}\,\mathbb{E}_{x\sim pdata}\,logD(x)+\frac{1}{2}\,\mathbb{E}_{z}\,log(1-D(G(z))) \tag{4.28}$$

Conforme mencionado anteriormente, o *framework GAN* inclui o modelo gerador (G) e o modelo discriminador (D). Ambos os modelos implementados nesse módulo são estruturas de *Deep Neural Networks* (*DNNs*), que são compostas de diversas camadas

ocultas [142]. As *DNNs* permitem extrair hierarquicamente a abstração do conhecimento e as características e os padrões de aprendizagem dos dados brutos da rede. Além disso, elas fornecem mais representação dos dados de entrada para melhorar a taxa de detecção de ataques complexos em um ambiente de rede de alta velocidade, principalmente os ataques de *DDoS* [143, 144].

O gerador (G) foi implementado com múltiplas camadas totalmente conectadas (*fully-connected*) (Densa) e uma camada de saída linear. A Tabela 3 mostra a estrutura do modelo G. Também foram implementadas várias camadas totalmente conectadas para o discriminador (D) e uma camada sigmoide para a camada de saída, em que a saída é a classificação do comportamento da rede no intervalo de análise. Dessa forma, o método proposto classifica o tráfego de rede como normal ou ataque *DDoS*. Além disso, o discriminador (D) durante o treinamento adversário, aprende o comportamento normal da rede, o que é vantajoso para detectar ataques *zero-day*. A Tabela 4 detalha as camadas do modelo D.

Tabela 3 – Estrutura do modelo do Gerador (G).

#	**Camada**	**Neurônios**	**Função de ativação**
1	*Fully-connected* (Densa)	6	ReLU
2	*Fully-connected* (Densa)	10	ReLU
3	*Fully-connected* (Densa)	8	ReLU
4	*Fully-connected* (Densa)	6	Linear

Fonte: O próprio autor.

Tabela 4 – Estrutura do modelo do Discriminador (D).

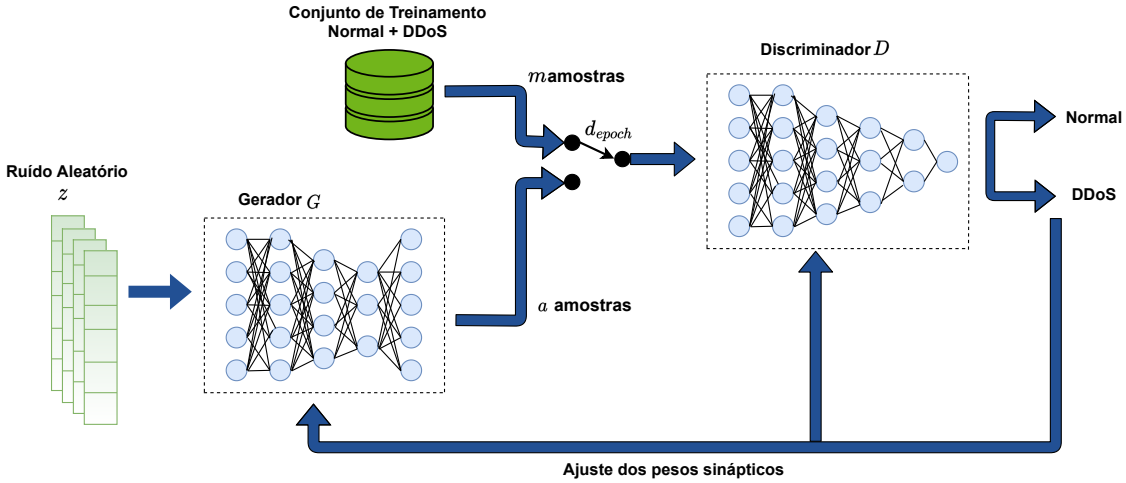
#	**Camada**	**Neurônios**	**Função de ativação**
1	*Fully-connected* (Densa)	12	ReLu
2	*Fully-connected* (Densa)	10	ReLu
3	*Fully-connected* (Densa)	6	ReLU
4	*Fully-connected* (Densa)	1	Sigmoid

Fonte: O próprio autor.

O fluxograma presente na Figura 18 ilustra o processo de treinamento do adversário utilizado. O primeiro passo é treinar o modelo D para d_{epoch}. Durante essa etapa, um *mini-batch* de m amostras do conjunto de dados de treinamento é amostrado aleatoriamente. O conjunto de dados de treinamento consiste em tráfego normal e ataques de *DDoS*. Então, o modelo G gera exemplos adversários a a partir do ruído aleatório z. Os conjuntos m e a são usados para treinar o modelo D. Os pesos sinápticos dos modelos D e G são atualizados de acordo com seus gradiente estocásticos, definidos na Equação (4.29) e na Equação (4.30), respectivamente.

$$\nabla_{\theta_D} \frac{1}{m} \sum_{i=1}^{m} \left[logD\left(m^{(i)}\right) + log\left(1 - D(G(z^{(i)})\right)\right] \tag{4.29}$$

$$\nabla_{\theta_G} \frac{1}{z} \sum_{i=1}^{z} log\left(1 - D(G(z^{(i)})\right) \tag{4.30}$$



Fonte: O próprio autor.

Figura 18 – Estrutura de treinamento da GAN.

O processo adversário de geração e treinamento é repetido até que o modelo D possa detectar as amostras falsas ou um número máximo de iterações de treinamento t. Esse processo é ilustrado a seguir no Algoritmo 3.

Algorithm 3 Etapas de treinamento do Adversário da GAN

Require: Modelo do gerador G; Modelo discriminador D; Conjunto de dados de treinamento

1: **while** t iterações de treinamento ou condição de parada não atendida **do**
2: **for** d_{epoch} **do**
3: *mini-batch* de tamanho m do conjunto de dados de treinamento
4: Gera exemplos adversários a a partir de G
5: Atualiza o discriminador pelo seu gradiente estocástico
6: Seleciona z amostras do ruído $z \sim p_z$
7: Atualiza o gerador pelo seu gradiente estocástico
8: **return** G e D

4.3.2 Análise de complexidade computacional da *GAN*

Na fase de operação da rede, após o seu treinamento, o discriminador (D) será utilizado no sistema para a detecção dos eventos anômalos. Dessa maneira, será avaliada a

complexidade computacional do discriminador na fase de operação, uma vez que o custo computacional na fase de treinamento não é afetado na fase de operação.

O discriminador é uma *DNN* composta de cinco camadas densas, incluindo a camada de entrada, em que todos os neurônios de uma camada são conectados com os neurônios da próxima camada. Considerando a o número de neurônios da camada 1; b o da camada 2, c da camada 3; d o número de neurônios na camada 4; e e o da camada 5.

Como existem cinco camadas, são necessárias quatro matrizes para representar os pesos sinápticos. As matrizes são representadas W_{ba}, W_{cb}, W_{dc} e W_{ed}, onde W_{ba} é uma matriz com b linhas e a colunas (W_{ba} contém os pesos sinápticos que vão da camada a para a camada b, e assim para as demais camadas).

Assumindo $\mathbf{x}$ como o vetor de entrada de tamanho a para a propagação da camada a para a camada b, tem-se:

$$\mathbf{S}_b = \mathbf{W}_{ba} \times \mathbf{x}_a \tag{4.31}$$

em que S_b é o resultado da multiplicação do vetor entrada pelos pesos sinápticos, com essa operação tendo complexidade de tempo de $\mathcal{O}(b \times a)$. Em seguida, é aplicada a função de ativação:

$$\mathbf{I}_b = f(\mathbf{S}_b) \tag{4.32}$$

que tem complexidade de tempo $\mathcal{O}(b)$. Então, a complexidade de tempo total de propagação de sinal da camada a para a camada b é de:

$$\mathcal{O}(b \times a + b) = \mathcal{O}(b \times (a + 1)) = \mathcal{O}(b \times a) \tag{4.33}$$

Aplicando a mesma lógica nas demais camadas, para propagar o sinal $b \rightarrow c$, tem-se $\mathcal{O}(c \times b)$; $c \rightarrow d$, tem-se $\mathcal{O}(d \times c)$; e $d \rightarrow e$ tem-se $\mathcal{O}(e \times d)$.

A complexidade total de tempo do discriminador será dada por:

$$\mathcal{O}(b \times a + c \times b + d \times c + e \times d) = \mathcal{O}(ab + bc + cd + de) \tag{4.34}$$

4.4 Considerações sobre o capítulo

Neste capítulo, foi apresentado o sistema para a detecção e a mitigação de anomalias em redes *SDN* aplicando redes neurais profundas. Foram apresentadas e detalhadas as soluções desenvolvidas e que foram implementadas em um arquitetura modular com funções bem definidas.

A primeira solução apresentada utiliza a rede neural profunda *LSTM* e a lógica Fuzzy para a detecção e mitigação de anomalias em redes *SDN*. A *LSTM* foi aplicada com o objetivo de aprender e atrair padrões para prever o comportamento normal da rede e, em conjunto Fuzzy, detectar ataques *DDoS* e *Portscan*.

A segunda solução que foi apresentada neste capítulo utiliza o *framework GAN* para detecção de ataques de *DDoS* aplicando o treinamento adversário para torar o sistema menos sensível a ataques adversário.

No próximo capítulo, serão avaliadas em diferentes cenários as soluções desenvolvidas e comparados os resultados obtidos com outros métodos presentes da literatura.

5 Resultados

Neste capítulos, são avaliadas todas as soluções desenvolvidas nesta tese. Essas soluções foram desenvolvidas utilizando a linguagem de programação Python com as bibliotecas Keras e [112] e TensorFlow [145] para o desenvolvimento de aplicações de *Deep Learning*. Os experimentos foram realizados em um ambiente com a seguinte configuração: Intel Core i5 2,21 GHz, 8 GB de memória RAM, sistema Windows 10. Além disso, as soluções desenvolvidas foram avaliadas em diferentes cenários e comparadas a outras soluções presentes na literatura.

As seções a seguir descrevem as métricas utilizadas nas avaliações, os resultados obtidos em cada uma das soluções e a comparação com outros métodos que foram empregados em trabalhos anteriores para detecção de ataques em redes *SDN*. Na seção 5.2 foram avaliados e analisados os resultados obtidos com a *LSTM* e a lógica Fuzzy; na seção 5.3 o resultados obtidos com a *GAN*; e na seção 5.4, a comparação entre os desempenhos das duas soluções.

5.1 Métricas e teste de avaliação

Os testes aplicados têm como objetivo verificar a eficiência do sistema proposto com relação aos módulos que o compõem: previsão, detecção e mitigação. Os resultados de desempenho dos modelos propostos foram analisados empregando as seguintes métricas estatísticas [146]: *precision*, *recall*, *F1 Score* e taxa de falso-positivo, descritas a seguir.

1. *precision*: apresenta o percentual de intervalos classificados como anomalias que, de fato, são anômalos. Nos sistemas de detecção de anomalias, essa métrica é utilizada para penalizar as situações em que um tráfego legítimo é classificado como ataque.

2. *recall*: mede quão efetivo é o modelo em classificar os intervalos anômalos com relação a todos os intervalos. Essa métrica é utilizada para medir a capacidade dos sistemas de detecção de anomalias em identificar os ataques e considera os falsos-negativos mais prejudiciais que os falsos-positivos, ou seja, um ataque não detectado é considerado mais prejudicial que um tráfego legítimo detectado como ataque.

3. *F1 score*: é a média harmônica entre as métricas *precision* e *recall*. Essa métrica é uma maneira simples de encontrar o equilíbrio entre as métricas de *precision* e *recall*, ou seja, avalia a capacidade geral do sistema de detecção de anomalias: quanto mais os valores estiverem próximos a 1, melhor é o desempenho do sistema.

4. taxa de falso-positivo (*False Positive Rate - FPR*): expressa um erro na classificação, em que o tráfego foi identificado como uma anomalia, mas na verdade é legítimo.

Os sistemas de detecção de anomalias devem apresentar baixas taxas de falsos-positivos, pois uma taxa de falsos alarmes sobrecarrega o sistema, prejudicando usuários legítimos.

Essas métricas podem ser calculadas facilmente por meio das seguintes equações:

$$\text{precision} = \frac{TP}{(TP + FP)} \tag{5.1}$$

$$\text{recall} = \frac{TP}{(TP + FN)} \tag{5.2}$$

$$\text{F1 score} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5.3}$$

$$\text{FPR} = \frac{FP}{(FP + FN)} \tag{5.4}$$

em que *TP* (*true-positive*), *TN* (*true-negative*), *FP* (*false-positive*) e *FN* (*False Negative*) significam verdadeiro-positivo, verdadeiro-negativo, falso-positivo e falso-negativo, respectivamente.

A acurácia, definida na Equação (5.5) é uma métrica amplamente utilizada em trabalhos de detecção de anomalias, porém pode levar a resultados tendenciosos quando a base de dados está desbalanceada, que é o caso dos dados aplicados neste trabalho. A base contém mais amostras normais que anômalas e o sistema pode classificar todas as amostras normais corretamente e classificar de modo errado as amostras anômalas, fornecendo um resultado tendencioso. A métrica *precision* pode ser aplicada para resolver esse resultado tendencioso e enfatizar a classificação correta das amostras anômalas.

$$\text{Acurácia} = \frac{TP + TN}{(TP + TN + FP + FN)} \tag{5.5}$$

Combinando as taxas de *TP* e *FP* é obtida uma análise visual da capacidade do sistema na detecção de anomalias. Essa análise visual é chamada *Receiver Operating Characteristics* (*ROC*) [146]. O cálculo da área sobre a curva (*AUC*, do inglês *Area Under the Curve*) permite quantificar a eficiência entre vários classificadores. Aquele que apresentar o maior valor para essa métrica é o mais apto a classificar as amostras.

O desempenho da fase de mitigação das anomalias detectadas foi avaliado pela taxa correta dos pacotes anômalos descartados e também foi aplicado o teste estatístico de McNemar. O Teste de McNemar é um teste não-paramétrico e sua aplicação é feita por meio de amostras pareadas e dados nominais. É aplicado em amostras em que o indivíduo tem dois comportamentos, um antes e outro depois de um tratamento [147].

Para testar a significância de qualquer mudança, é verificado se as frequências marginais são iguais ou não. Desse modo, é necessário construir uma tabela de contingência 2x2 para representar os dois comportamentos (o anômalo e o normal). Uma tabela de contingência registra a frequência das observações de múltiplas variáveis qualitativas. As

linhas e as colunas dessa tabela correspondem a tais variáveis e os valores nas células representam as frequências observadas para as variáveis. Na Tabela 5, é ilustrado um exemplo genérico de uma tabela de contingência 2x2 que apresenta o resultado de um tratamento genérico em uma amostra de n indivíduos e as classes A e B. Com base nessa tabela, podem ser definidas quatro situações.

(a) quantidade de amostras que eram A e continuaram a ser A após o tratamento;

(b) quantidade de amostras que eram A e passaram a ser B após o tratamento;

(c) quantidade de amostras que eram B e passaram a ser A após o tratamento;

(d) quantidade de amostras que eram B e continuaram a ser B após o tratamento.

Tabela 5 – Tabela de contingência 2x2 genérica.

	A depois do tratamento	B depois do tratamento	Total da linha
A antes do tratamento	a	b	a+b
B antes do tratamento	c	d	c+d
Total da coluna	a+c	b+d	n

Fonte: Adaptada de [147].

A hipótese nula indica que as probabilidades para cada resultado são iguais, isto é, não houve alteração nas frequências marginais. A hipótese nula e a hipótese alternativa são representadas, respectivamente, por:

$$H_0 : b = c$$
$$H_1 : b \neq c$$

A fórmula do teste de McNemar se origina da fórmula do Qui-quadrado:

$$\chi^2 = \frac{(b-c)^2}{b+c} \tag{5.6}$$

χ^2 tem uma distribuição qui-quadrado com 1 grau de liberdade. Se o resultado de χ^2 é significativo, isto é, se as frequências marginais são significativamente diferentes umas das outras ($b \neq c$), a hipótese nula é rejeitada.

No processo de avaliação, o teste será aplicado para verificar os intervalos que eram anômalos antes da mitigação e passaram a ser normais após ela. Desse modo, poderá ser observado se o módulo presente na fase de mitigação foi efetivo em mitigar os ataques detectados.

5.2 Resultados Sistema 1: *Long Short-Term Memory* e Lógica Fuzzy para detecção e mitigação de anomalias em redes *SDN*

Para demonstrar a eficácia do sistema proposto, foram aplicados testes com base em cenários distintos. O ambiente de teste utilizado no primeiro cenário foi uma topologia de rede com 120 *hosts* e com períodos de ataque de *DDoS* e *Portscan*. No segundo cenário, foi avaliado o sistema usando o conjunto de dados públicos chamado CICDDoS 2019 [148] do *Canadian Institute for Cybersecurity*, que contém diferentes tipos de ataque de *DDoS* e de perfis de tráfego de redes *SDN* realistas.

5.2.1 Descrição dos cenários avaliados

Com o objetivo de avaliar o sistema proposto, foram analisados dois cenários para mensuração de testes quantitativos e qualitativos. No primeiro cenário, foram utilizados dados emulados gerados pelo grupo ORION de pesquisa em redes de computadores do Departamento de Ciência da Computação da Universidade Estadual de Londrina [1]. No segundo cenário, foram utilizados dados de uma base pública chamada *CICDDoS* 2019 [148], composta por diferentes tipos de ataque de *DDoS*.

O sistema faz a análise do comportamento do tráfego a cada segundo. Portanto, os fluxos de rede devem ser coletados no mesmo intervalo. Considerando essa análise, no cenário 1 foi necessário emular o comportamento da rede utilizando o emulador de rede Mininet [149]. A principal característica desse emulador é a facilidade para prototipação de redes virtuais realistas com componentes *SDN*, tais como controladores, *hosts*, *links* e *switches*. A prototipação pode ser feita em apenas uma máquina virtual. Além disso, ela utiliza uma virtualização leve, que permite a criação de topologias personalizadas de código aberto e é amplamente aplicada no estado da arte de pesquisas e desenvolvimentos de soluções em ambientes *SDN*.

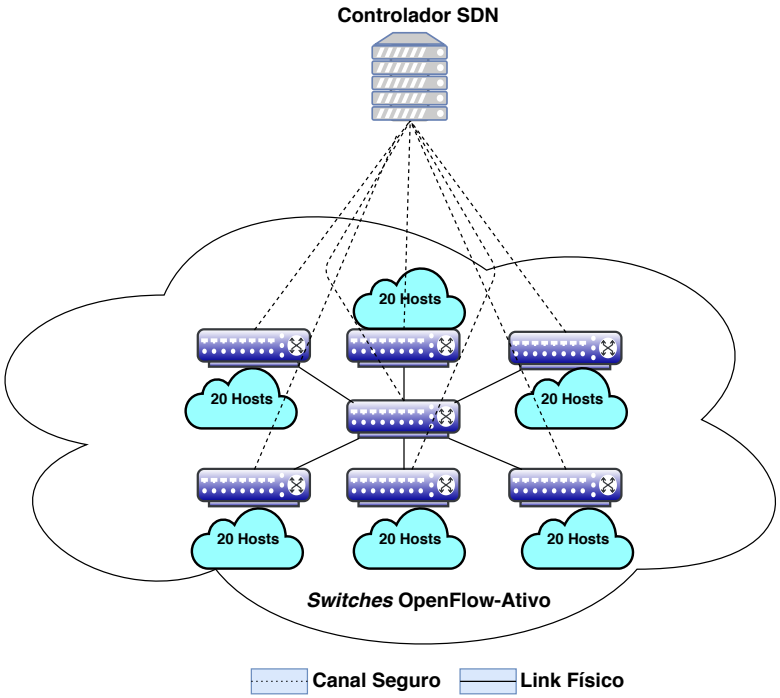
A geração de tráfego na rede emulada foi realizada por intermédio da ferramenta chamada Scapy [150], a qual permite que o cenário emulado opere com altas taxas de tráfego, criando um cenário mais próximo possível de um ambiente *SDN* real. O controlador de rede *SDN* utilizado foi o Floodlight [151], baseado na linguagem de programação Java e desenvolvido pela BigSwitch. Ele oferece suporte a uma ampla variedade de *switches* *OpenFlow* virtuais e físicos e pode lidar com redes mistas *OpenFlow* e não *OpenFlow*. Os atributos de fluxos utilizados foram: *bit*/s, pacotes/s, entropia de *IP* de origem, entropia de *IP* de destino, entropia de porta de origem e entropia de porta de destino.

A Figura 19 ilustra a topologia emulada no cenário 1. O primeiro cenário é formado por uma topologia em que seus elementos estão dispostos no formato de estrela. Essa

[1] <http://www.uel.br/grupos/orion/datasets.html>

topologia é composta de um *switch* central, conectado a outros seis *switches*. Cada sub-rede contém 20 *hosts*, totalizando 120 *hosts*. Foram emulados dois dias de 24 horas, cada dia com 86400 amostras. O primeiro dia de emulação contém apenas amostras do comportamento normal da rede e foi empregado na fase de treinamento da *LSTM*, visto que utilizou-se uma abordagem de treinamento semi-supervisionada. O segundo dia emulado foi utilizado para avaliar o desempenho de operação do sistema na detecção e na mitigação de ataques. Ao longo da emulação, foram realizados dois ataques com diferentes intensidades e duração: um ataque de *DDoS* e um ataque de *Portscan*. As informações relacionadas aos parâmetros utilizados nos ataques estão detalhadas na Tabela 6.

No segundo cenário, foi utilizada a base de dados pública *CICDDoS* 2019 [148]. Esse conjunto de dados é composto de dois dias: um de treino e outro de teste. O conjunto de treinamento é composto de 12 diferentes tipos de ataque de *DDoS*, sendo eles: *NTP*, *DNS*, *LDAP*, *MSSQL*, *NetBIOS*, *SNMP*, *SSDP*, *UDP*, *UDP-Lag*, *WebDDoS* (*ARME*), *SYN* e *TFTP*. O segundo dia, o dia de teste, contém seis tipos de ataques de *DDoS*, sendo eles: *NetBios*, *LDAP*, *MSSQL*, *UDP*, *UDP-Lag* e *SYN*. As dimensões de fluxos utilizadas foram as mesmas do cenário 1.



Fonte: O próprio autor.

Figura 19 – Topologia de rede emulada no cenário 1 utilizando o Mininet.

5.2.2 Avaliação de hiperparâmetros

Nesta seção, encontram-se os resultados da avaliação dos parâmetros empregados na construção do sistema proposto utilizando as bibliotecas para o desenvolvimento de aplicações de *Deep Learning*: Keras [112] e TensorFlow [145]. Os parâmetros empregados no

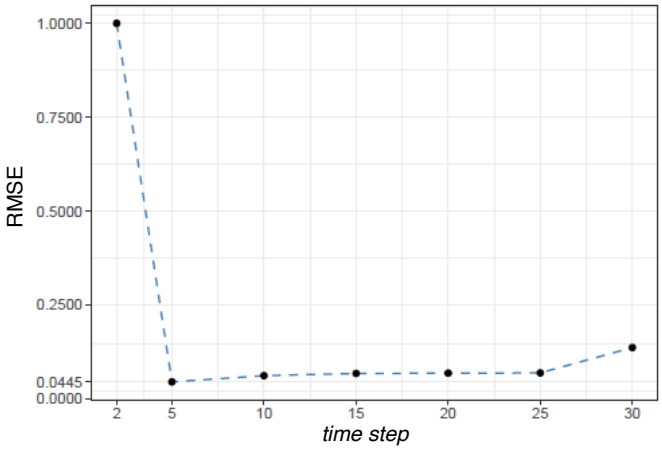
Tabela 6 – Informações sobre os parâmetros utilizados nos ataques do cenário 1.

Tipo de ataque	**Parâmetros**
DDoS	Atacantes: 16 IPs atacantes: 10.0.0.80 - 10.0.0.95 IP vítima: 10.0.0.50 Horário: 9:35 - 10:45
Portscan	IP atacante: 10.0.0.10 IP vítima: 10.0.0.24 Portas: 1 - 20000 Tempo entre os pacotes: 0.15 segundos Horário: 14:30 - 15:30

Fonte: O próprio Autor.

treinamento foram definidos como a função de perda *RMSE*, que é uma função clássica utilizada em tarefas de regressão. Além disso, foi aplicado o algoritmo *Adam* [152], que otimiza a taxa de aprendizagem das redes neurais profundas de maneira adaptativa.

O primeiro parâmetro avaliado foi o tamanho do *time step* empregado pela rede *LSTM* na fase de previsão do tráfego. Os valores utilizados para o teste compreendem entre 2 e 30 amostras anteriores do tráfego coletado. O *RMSE* foi calculado para determinar o melhor tamanho do *time step*. O *RMSE* é utilizado para calcular a diferença entre o valor predito e o valor real: valores próximos de zero indicam que o tráfego previsto é próximo do real. O gráfico presente na Figura 20 ilustra os valores de *RMSE* obtidos para cada um dos valores avaliados. O tamanho do *time step* que apresentou o melhor resultado foi igual a 5, com um valor de *RMSE* de 0,0445.

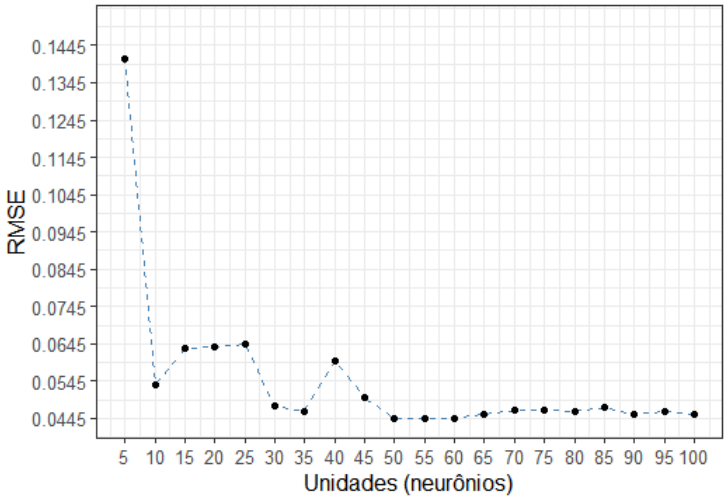


Fonte: O próprio autor.

Figura 20 – Tamanho do *time step* utilizado pela rede *LSTM* para a previsão do tráfego de rede.

O próximo passo foi definir o número de unidades ocultas. A avaliação desse parâmetro empregou o intervalo de 5 a 100 unidades e, para cada valor, o *RMSE* foi avaliado. O

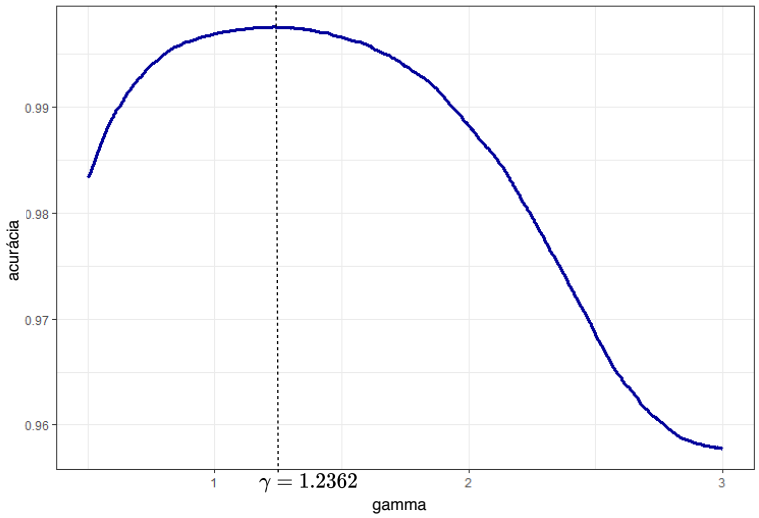
número de unidades que apresentou o melhor resultado foi 50. Após esse resultado, não houve melhora significante. Na Figura 21, está presente o gráfico com os resultados obtidos para cada valor de unidade e o seu respectivo valor de *RMSE*.



Fonte: O próprio autor.

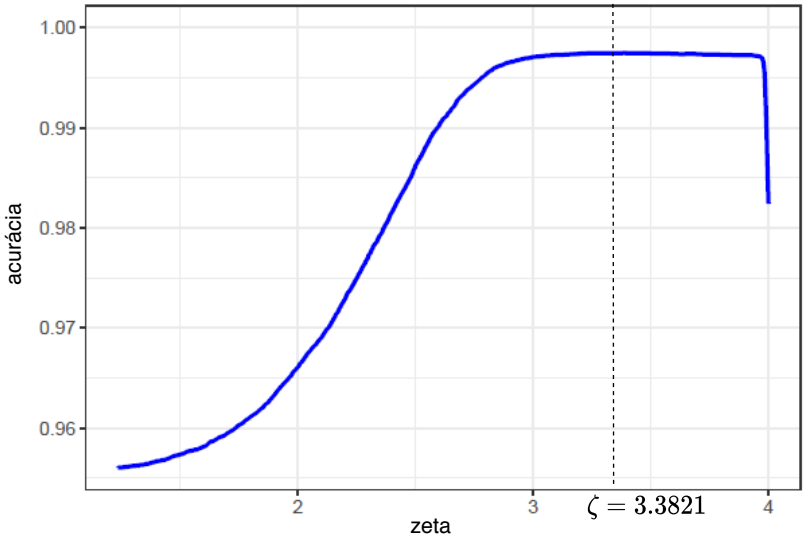
Figura 21 – Avaliação do número de unidades ocultas utilizadas na arquitetura da *LSTM*.

Os gráficos presentes na Figura 22 e na Figura 23 ilustram a avaliação dos valores de γ e ζ para encontrar o ponto de corte mais adequado da soma do *score* de anomalia. Os valores foram definidos variando γ e ζ e calculando a acurácia. O valor final de γ foi definido como $argmax_{\gamma}(accuracy_{\gamma})$ e o valor de ζ foi definido como $argmax_{\zeta}(accuracy_{\zeta})$. Os valores dos *scores* para γ e ζ foram definidos como 1,2362 e 3,3821, respectivamente. A Tabela 7 sumariza os parâmetros utilizados.



Fonte: O próprio autor.

Figura 22 – Avaliação do parâmetro gamma.



Fonte: O próprio Autor.

Figura 23 – Avaliação do parâmetro zeta.

Tabela 7 – Informações dos parâmetros e hiperparâmetros utilizados.

Parâmetro	**Descrição**	**Valor**
Função de perda	Função que compara os valores de saída esperados e previstos	RMSE
Otimizador	Algoritmo para atualização dos pesos e taxa de aprendizagem	Adam
Time step	Número de amostras utilizadas como entrada da rede	5
Unidades ocultas	Quantidade de neurônios presentes na camada oculta	50
γ	Limiar de anomalia para detecção de PortScan	1.2362
ζ	Limiar de anomalia para detecção de DDoS	3.3821

Fonte: O próprio autor.

5.2.3 Avaliação do Cenário 1: base emulada grupo ORION

Com o objetivo de realizar uma análise de desempenho, foi comparado o sistema proposto neste trabalho com outros cinco métodos que já foram aplicados para a detecção de anomalias em redes *SDN*. O primeiro método é o *k-Nearest Neighbor* (*kNN*) [153], um classificador supervisionado com baixo custo computacional utilizado para detecção de eventos maliciosos em *datacenter*. O segundo método é a *Multilayer Perceptron* (*MLP*) [143], uma rede neural artificial aplicada na detecção de ataques de *DDoS*. O terceiro método é baseado na técnica de classificação *Support Vector Machine* (*SVM*) [154], aplicada para a detecção de ataques de *flooding*. O quarto é um método heurístico presente na literatura chamado *Particle Swarm Optimization for Digital Signature* (*PSO-DS*) [130], que emprega uma técnica não supervisionada para a detecção de ataques de *DDoS* e *Portscan* em redes *SDN*. Por fim, o método de *Deep Learning LSTM-2* [52], empregado para a detecção de ataques de *DDoS* em ambientes *SDN*.

Para melhorar a comparação entre os métodos, nas abordagens supervisionadas (*kNN*, *SVM*, *MLP* e *LSTM-2*), empregou-se um conjunto de dados de treinamento que representa

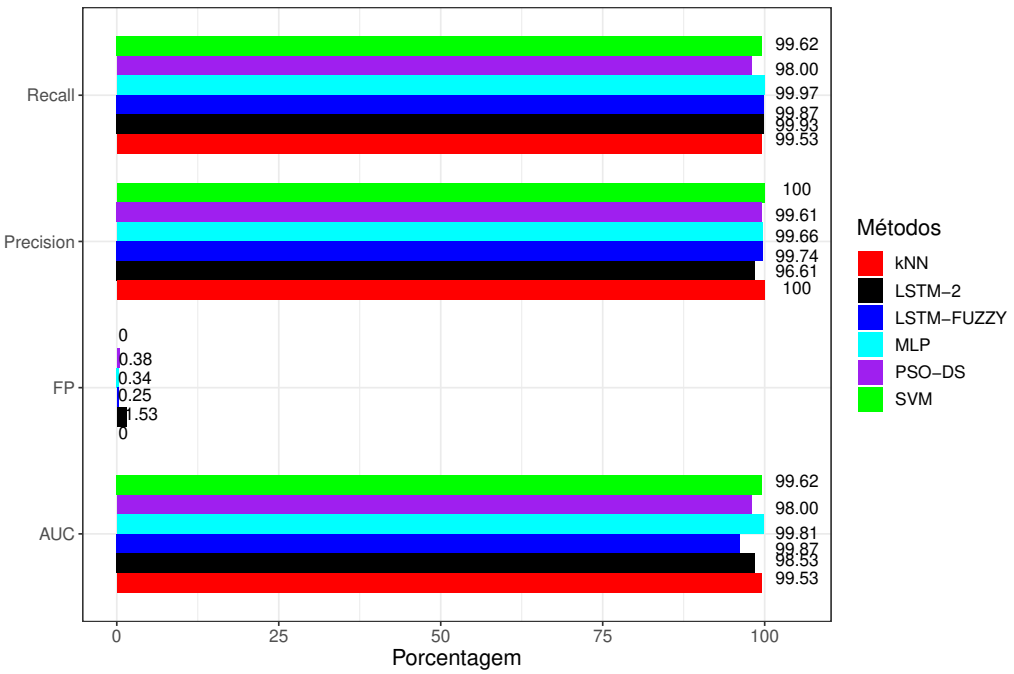
um dia de coleta de tráfego da rede. Esse dia consiste em por tráfego normal e ataques de *DDoS* e *Portscan*. As informações sobre a quantidade de amostras para cada tipo de tráfego estão ilustradas na Tabela 8.

Tabela 8 – Informações da quantidade de amostras para cada tipo de tráfego da rede.

Tipo de tráfego	Quantidade de amostras
Normal	78420
DDoS	4380
Portscan	3600
Total	86400

Fonte: O próprio autor.

Uma análise detalhada é ilustrada na Figura 24, em que são apresentados os resultados das métricas dos modelos comparados. Pode-se notar que a LSTM-FUZZY apresentou uma taxa de falso-positivo baixa, obtendo um valor de 0.25%. O método *LSTM-2* apresentou a maior taxa de falso-positivo, com 1.53%. Por outro lado, os métodos *SVM* e *kNN* não apresentaram taxa de falso-positivo. Com relação às métricas de *recall* e *precision*, todos os métodos apresentaram valores superiores a 98%. O método que obteve o pior desempenho entre os demais foi o *PSO-DS*. Nenhum dos métodos obteve melhor desempenho em todas as métricas avaliadas.

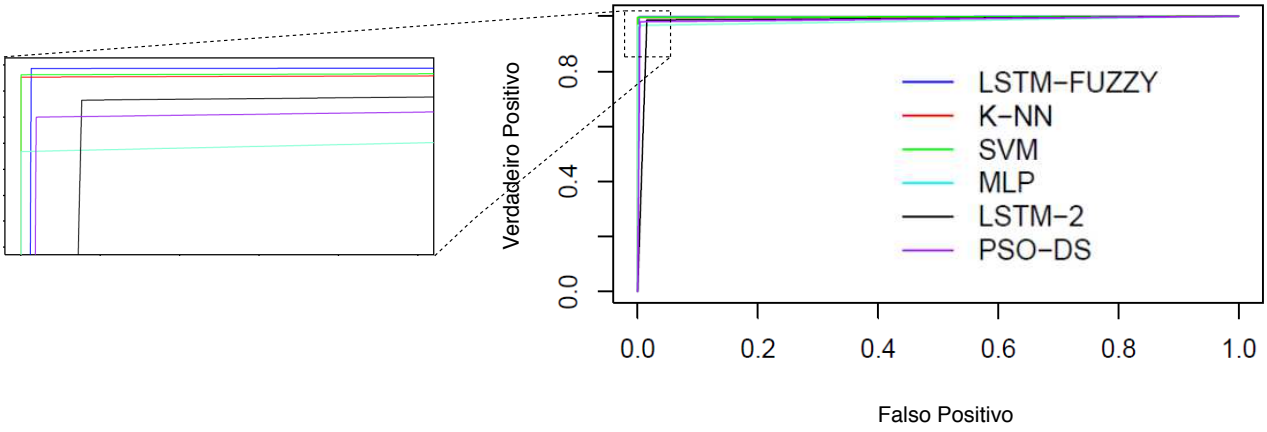


Fonte: O próprio autor.

Figura 24 – Resultados obtidos do sistema proposto e dos métodos comparados no primeiro cenário das métricas avaliadas.

A Figura 25 apresenta as curvas *ROC*, uma comparação visual entre os métodos. Por meio da curva *ROC*, é possível determinar qual dos métodos apresenta a aptidão mais

adequada para a detecção de anomalias. Analisando os resultados obtidos, é notável que método o LSTM-FUZZY teve o melhor desempenho entre os comparados, com um valor de *AUC* de 99.87%, resultando na maior taxa de verdadeiro-positivo e na menor taxa de falso-positivo.



Fonte: O próprio autor.

Figura 25 – Curvas *ROC* apresentadas pelos métodos comparados no cenário 1.

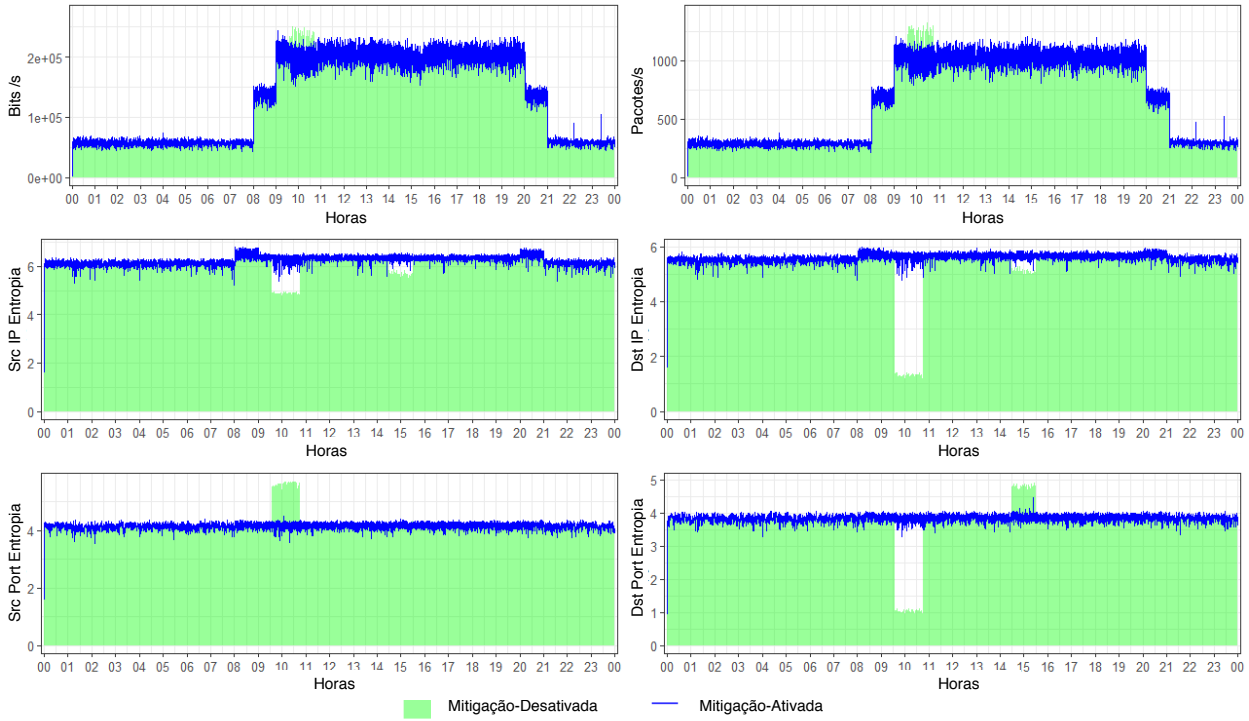
5.2.3.1 Mitigação dos ataques no cenário 1: base emulada grupo ORION

Com base nos alarmes gerados pelo processo de classificação do método LSTM-FUZZY, foram aplicadas as políticas de mitigação. A Figura 26 apresenta os atributos do tráfego sem a aplicação da mitigação em verde e o tráfego após a aplicação da mitigação em azul. No período entre 9:45:00 e 10:35:00, tem-se a ocorrência de um ataque de *DDoS*, sendo notável o aumento das taxas de pacotes e *bits* quando o módulo de mitigação está desabilitado. No entanto, com a ativação do módulo, o tráfego tende a voltar à sua normalidade devido aos descartes dos pacotes anômalos. O período do ataque de *Portscan*, entre 14:30 e 15:30, provoca pequenas alterações no comportamento do tráfego, mas, com a aplicação da mitigação, os atributos afetados também voltam à sua normalidade.

Nesse cenário, foi aplicada a análise da mitigação por meio do teste de McNemar e das taxas de pacotes descartados. O nível de significância para esse teste foi de $\alpha = 5\%$. A Tabela 9 fornece as informações sobre a classificação do tráfego entre anômalo e normal, antes e após o processo de mitigação. Essa tabela fornece também informações sobre quatro situações, descritas a seguir:

(i) intervalos de análise do tráfego que eram normais antes da mitigação e depois dela continuaram normais [normal (antes) – normal (depois)];

(ii) intervalos de análise do tráfego que eram normais antes da mitigação e depois dela passaram a ser classificados pelo sistema como anômalos [normal (antes) – anômalo (depois)];



Fonte: O próprio autor.

Figura 26 – Análise do tráfego com o módulo de mitigação desativado e ativado na ocorrência de ataques de *DDoS* e *Portscan*.

(iii) intervalos de análise do tráfego que eram anômalos antes da mitigação e depois dela passaram a ser classificados pelo sistema como normais [anômalo (antes) – normal (depois)];

(iv) intervalos de análise do tráfego que eram anômalos antes da mitigação e depois dela continuaram anômalos [anômalo (antes) – anômalo (depois)].

Aplicando o teste nas informações da Tabela 9, os resultados de *p-valor* ficaram abaixo de $2.2 \times e^{-16}$, que é menor que o valor de α. Logo, a hipótese nula foi rejeitada, indicando que houve diferença nas frequências e que a mitigação foi eficaz em minimizar os efeitos das ameaças. Além disso, a taxa de pacotes anômalos descartados pelo sistema foi de 99.88%, demonstrando que praticamente todos os pacotes anômalos foram descartados.

Tabela 9 – Tabela de contingência aplicada para a avaliação da mitigação no cenário 1.

	normal (depois)	**anômalo (depois)**
normal (antes)	78323	86
anômalo (antes)	7728	262

Fonte: O próprio autor.

5.2.4 Avaliação do Cenário 2: base de dados CICDDoS 2019

O objetivo desse cenário é avaliar o módulo de detecção do sistema aplicando diferentes tipos de ataque de *DDoS*. Conforme mencionado anteriormente, a base de dados CICDDoS 2019 [148], desenvolvida pelo *Canadian Institute for Cybersecurity*, é formada por dois dias (treino e teste). O dia de treino é composto de 12 tipos de ataques *DDoS* e o dia de teste contém seis tipos de ataque de *DDoS*.

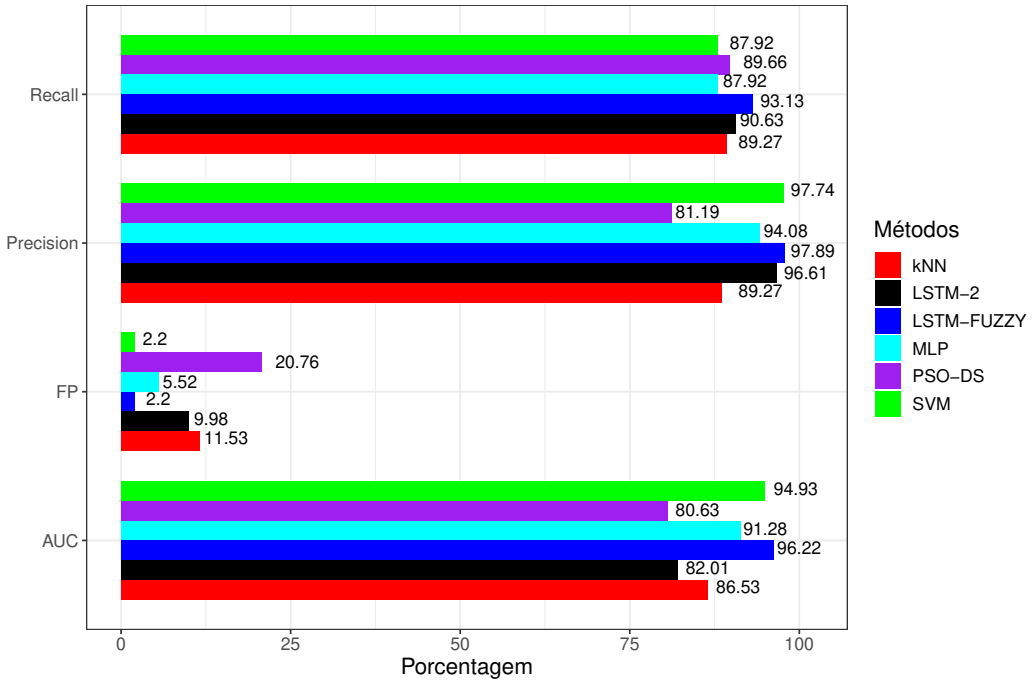
O sistema proposto neste trabalho realiza a análise do tráfego a cada segundo. Portanto, foi necessário fazer um pré-processamento da base de dados CICDDoS 2019 para separar as amostras de fluxos em intervalos de 1 segundo. Com isso, notou-se que todos os intervalos eram compostos apenas de amostras anômalas. Para resolver esse problema, antes do processo de separação em intervalos de 1 segundo, foram separados os fluxos em anômalos e normais.

Porém, a proporção das amostras de fluxos contendo ataques de *DDoS* é superior aos dados normais devido às características dos ataques. Esse não é um problema para o método LSTM-FUZZY, pois, na fase de treinamento, o método apenas utiliza as amostras normais para caracterizar o tráfego, mas pode gerar um *overfitting* para os métodos *SVM*, *kNN*, *MLP* e *LSTM-2*, que aplicam uma abordagem supervisionada na fase de treinamento. Para manter as características e a representatividade dos dados aplicados na fase de treinamento, a solução adotada para resolver esse problema foi amostrar aleatoriamente os fluxos para cada tipo de ataque. A fim de garantir a representatividade do conjunto de dados utilizado, para cada classe de ataque, foi selecionada uma proporção de cinco vezes o total de fluxos normais. O conjunto de treinamento foi reduzido, porém, manteve uma quantidade de amostras suficiente para o processo de treinamento.

Conforme realizado no primeiro cenário, a eficiência do método LSTM-FUZY foi comparada à de métodos clássicos: *kNN*, *SVM*, *MLP*, *LSTM-2* e *PSO-DS*. A Figura 27 ilustra os resultados das métricas obtidas por cada um desses métodos. Com relação à métrica *recall*, pode-se notar que o método LSTM-FUZZY teve um desempenho superior em relação aos demais métodos, obtendo um valor de 93.13%, seguido por *LSTM-2*, *PSO-DS*, *kNN*, *MLP* e *SVM*, que obtiveram as taxas de 90,53%, 89,66 %, 89,27%, 87,92% e 87,92%, respectivamente. A métrica avaliada na sequência foi a *precision*. O método LSTM-FUZZY novamente atingiu o melhor resultado, com uma taxa de 97,89%, os demais foram *SVM*, *LSTM-2*, *MLP*, *kNN* e *PSO-DS*, obtendo as taxas de 97,74%, 96,61%, 94,98%, 89,27% e 81,19%, respectivamente.

Já em comparação à taxa de falso-positivo, o método LSTM-FUZZY e a *SVM* obtiveram o mesmo valor: 2,2%, que pode ser considerado um bom resultado. Em seguida, ficaram os métodos *MLP*, *LSTM-2*, *kNN* e *PSO-DS*, com os respectivos valores: 5,52%, 9,98%, 11,53% e 20,76%. O sistema proposto apresentou resultados superiores aos demais métodos comparados, exceto para a *SVM*, que obteve resultados semelhantes. Porém, ao

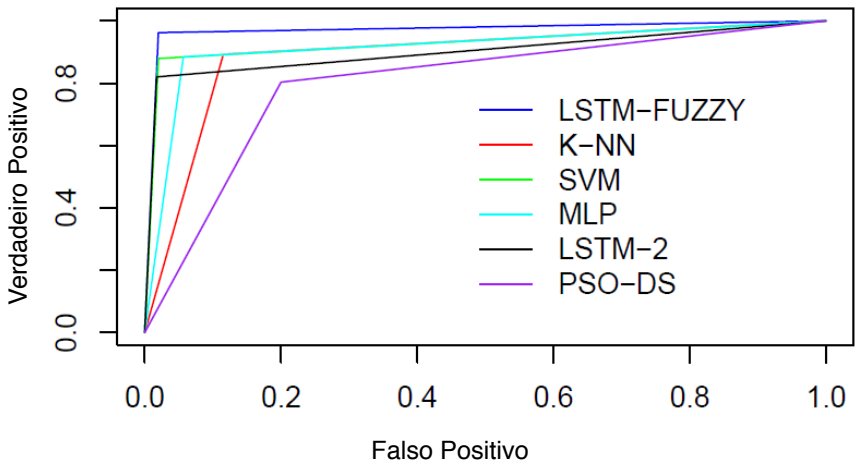
utilizar a curva *ROC*, foi possível observar com mais clareza a melhora entre os métodos comparados. Apesar dos resultados semelhantes, o desempenho aplicado pelo sistema proposto é uma melhoria significativa, visto que as redes de computadores atuais operam com *links* de altas taxas de transmissão. Ao longo de um dia de operação da rede, uma pequena porcentagem de ataques não detectados pode causar danos à sua operação. Por exemplo, em outubro de 2016, um ataque *DDoS* com 100 mil *hosts* maliciosos ultrapassou uma largura de banda de 1,2 *Tbps* [155]. Como nos resultados apresentados no primeiro cenário, o método LSTM-FUZZY teve uma média melhor do que os outros métodos comparados no segundo cenário, obtendo resultados de testes promissores, que o tornam um método eficiente na detecção de diferentes tipos de ataque de *DDoS*.



Fonte: O próprio Autor.

Figura 27 – Resultados obtidos do sistema proposto e dos métodos comparados no segundo cenário das métricas avaliadas.

Assim como no cenário anterior, a curva *ROC* foi utilizada para determinar qual método apresentou o melhor desempenho na detecção dos ataques. A Figura 28 apresenta a análise visual da curva *ROC*. Por meio da métrica *AUC*, é notável que o método LSTM-FUZZY foi o que mostrou melhor equilíbrio entre as taxas de verdadeiro-positivo e falso-positivo, obtendo um valor de 96,22%, seguido dos métodos *SVM*, *MLP*, *kNN*, *LSTM-2* e *PSO-DS*, com os seguintes valores, respectivamente: 94,93%, 91,28%, 86,53%, 82,01% e 80,63%.



Fonte: O próprio autor.

Figura 28 – Curvas *ROC* apresentadas pelos métodos comparados no cenário 2.

5.2.4.1 Mitigação dos ataques no cenário 2: base de dados CICDDoS 2019

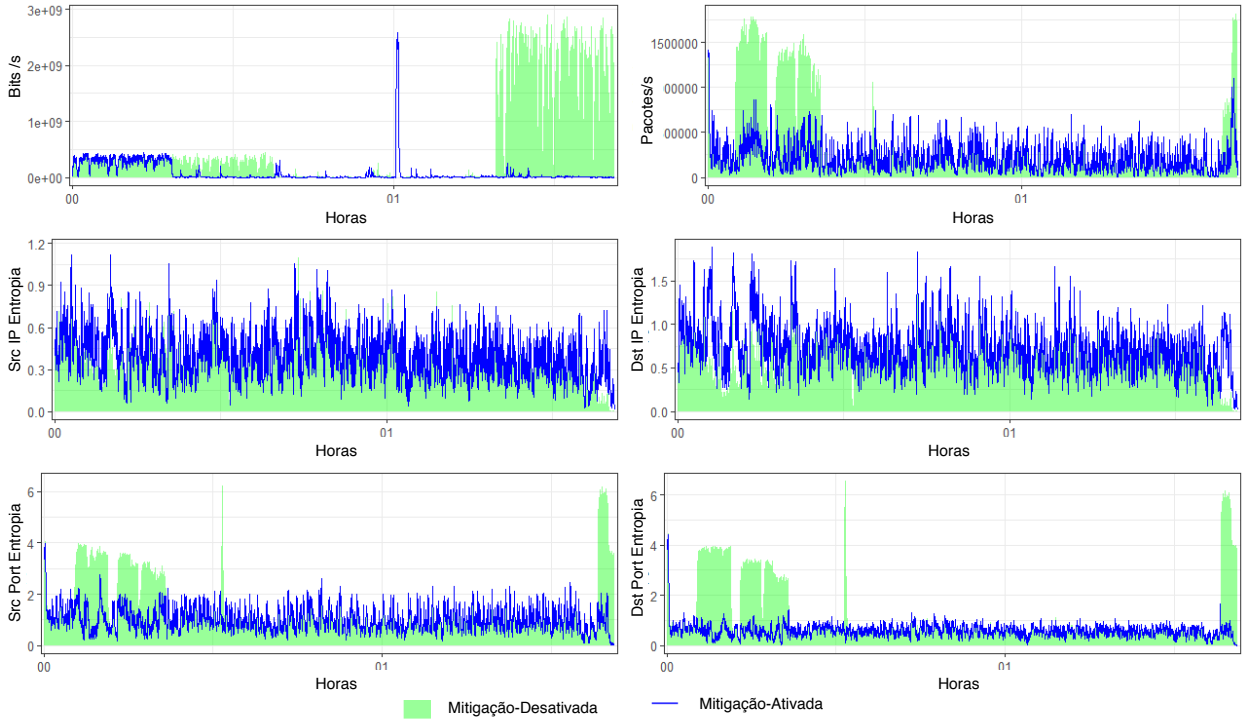
Nesse cenário, também foi avaliada a eficiência do módulo de mitigação do sistema proposto. A Figura 29 apresenta o comportamento do tráfego do dia de teste, comparando a ocorrência de ataques de *DDoS* com o módulo de mitigação desativado e, depois, seu comportamento quando a mitigação está ativada. O tráfego gerado sem a aplicação da política de mitigação é representado pela área em verde e a linha em azul mostra o tráfego depois de aplicada a mitigação contra os ataques de *DDoS*. Visualmente, é possível notar quando os ataques são mitigados, pois os valores dos atributos em análise retornam ao seu comportamento normal.

O teste de McNemar foi aplicado com o nível de significância de $\alpha = 5\%$ e a hipótese nula de que as frequências marginais são iguais. Aplicando o teste na Tabela 10 de contingência, o resultado do *p-valor* foi de $2,2 \times e^{-16}$, que é menor que o valor de α. Logo, a hipótese nula é rejeitada, o que indica que houve diferença nas frequências marginais e que a mitigação foi efetiva. Além disso, a taxa de pacotes anômalos descartados foi de 99.20%, demonstrando que a maioria dos pacotes anômalos foi mitigada.

Tabela 10 – Tabela de contingência aplicada para a avaliação da mitigação no cenário 2.

	normal (depois)	**anômalo (depois)**
normal (antes)	4811	136
anômalo (antes)	2175	68

Fonte: O próprio Autor.



Fonte: O próprio autor.

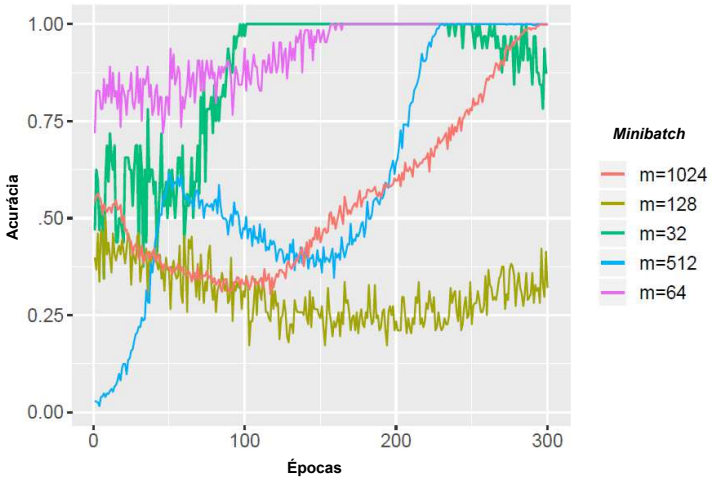
Figura 29 – Análise do tráfego com o módulo de mitigação desativado e ativado na base CICDDoS 2019.

5.3 Resultados Sistema 2: *Adversarial Deep Learning* para detecção e defesa contra ataques *DDoS* em ambientes *SDN*

Para avaliar o desempenho do sistema proposto, foram aplicados testes em diferentes cenários e os resultados foram comparados com os de outros sistemas presentes na literatura. O primeiro cenário de teste contém dados da emulação de uma rede *SDN* com alta taxa de transmissão e diversos dispositivos conectados, totalizando 128 *hosts*. Nesse cenário, há dois períodos de ataques *UDP DDoS* com diferentes intensidades. No segundo cenário, aplicamos o conjunto de dados público CICDDoS 2019 [148] do Canadian Institute for Cybersecurity, que contém 28 perfis normais e tipos recentes de ataques de *DDoS*.

5.3.1 Avaliação de hiperparâmetros

Durante a fase de treinamento do *framework GAN* foram avaliados os hiperparâmetros, sendo eles o tamanho do *mini-batch* e o número de d_{epoch} para treinar o discriminador antes de atualizar o gerador. Esses parâmetros foram avaliados para encontrar um ponto de equilíbrio entre o gerador e o discriminador. Além disso, as configurações dos parâmetros

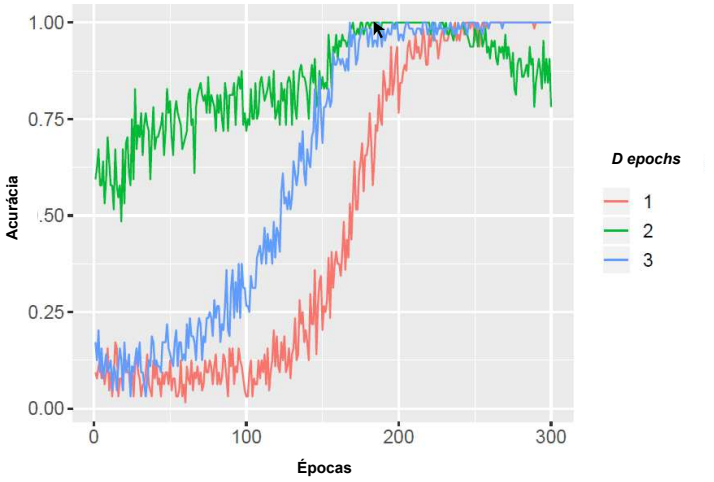


Fonte: O próprio Autor.

Figura 30 – Avaliação do tamanho de *mini-batch*.

de treinamento foram definidas como *Dropout* com taxa de 0,2 para evitar *overfitting* [113]; a função de perda *Binary Crossentropy*, que é uma função de perda clássica usada em tarefas de classificação binária; a taxa de aprendizado padrão igual a 0,001 definida em [112]; e o otimizador foi definido como *Adam* [152], que é um algoritmo de otimização de taxa de aprendizagem adaptativo para treinar redes neurais profundas.

Para avaliar o tamanho do *mini-batch*, foram aplicados os seguintes valores: 32, 64, 128, 256, 512 e 1024 amostras. O número de épocas e a taxa de acurácia foram avaliados para obter o valor de convergência. A Figura 30 ilustra os resultados desse teste. Um tamanho de *mini-batch* de 32 alcançou uma taxa de acurácia melhor, mas diminuiu após 257 épocas. Conforme observado, o melhor resultado obtido foi utilizando um tamanho de *mini-batch* de 64, no qual o número de épocas necessárias para a convergência foi 157.



Fonte: O próprio autor.

Figura 31 – Avaliação do hiperparâmetro d_{epoch}.

O próximo parâmetro foi o número de d_{epoch} para alternar e atualizar o gerador e o

discriminador. De acordo com as informações apresentadas em [29], valores pequenos podem alcançar melhores resultados. Dessa forma, a estimação do referido parâmetro d_{epoch} foi aplicada entre 1 e 3 épocas. Essa avaliação é detalhada na Figura 31. Conforme os resultados alcançados, o número de d_{epoch} que atingiu a convergência adequada foi igual a 2. A Tabela 11 sumariza os parâmetros utilizados e os seus respectivos valores.

Tabela 11 – Informações dos parâmetros e hiperparâmetros utilizados na GAN.

Parâmetro	**Descrição**	**Valor**
Função de perda	Função que compara os valores de saída esperados e previstos	Binary CrossEntropy
Otimizador	Algoritmo para a atualização dos pesos e a taxa de aprendizagem	Adam
mini-batch	Número de amostras apresentadas para a atualização dos pesos	64
d_{epoch}	Número de épocas para alternar e a atualizar o gerador e o discriminador	2

Fonte: O próprio autor.

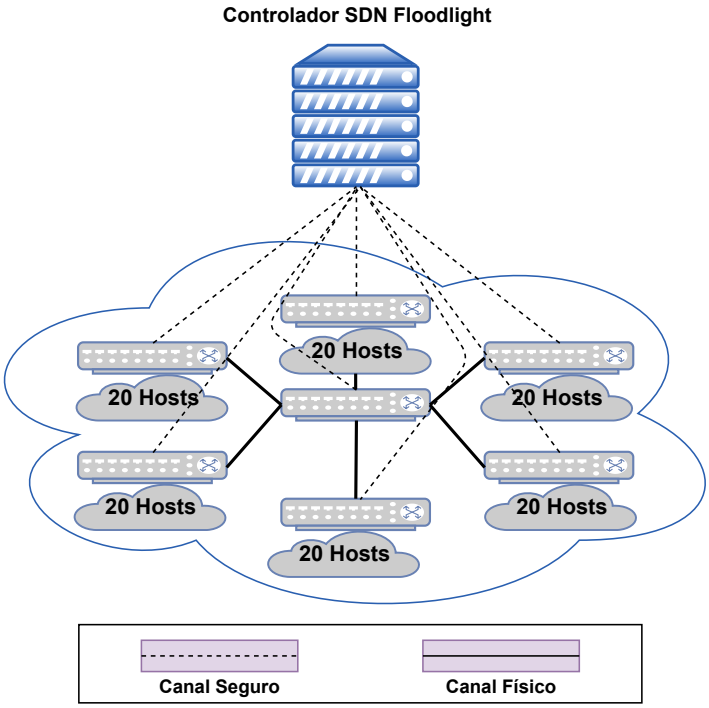
5.3.2 Resultados cenário 1: base de dados *DDoS* Orion

Nesse cenário, foi emulada uma topologia de rede por meio do emulador de rede Mininet [149], que é amplamente aplicado no desenvolvimento de soluções *SDN* devido à facilidade de implementação de ambientes *SDN* virtuais realistas compostos de controladores, *hosts*, *links* e *switches* em uma máquina virtual. O arranjo de rede emulado é uma topologia em estrela na qual seis *switches* são conectados a um *switch* central. Cada sub-rede contém 20 *hosts*, totalizando 120 *hosts*, conforme descrito na Figura 32. Além disso, foi utilizado o controlador *SDN* Floodlight [151], que é baseado na linguagem de programação Java e tem sido utilizado em diversas aplicações *SDN* como controlador de rede.

Nesse cenário, foi emulado o comportamento de uma rede com altas taxas de transmissão por 24 horas. Para injetar tráfego na rede emulada, foi utilizada a ferramenta chamada Scapy [150], aplicada na manipulação de pacotes para redes de computadores. Essa ferramenta permite que uma rede emulada possa ser semelhante a um cenário de rede real.

Dois ataques *UDP DDoS* foram realizados durante a fase de emulação com diferentes intensidades e tempo de duração. Esse tipo de ataque é o método *DDoS* comum usado por invasores contra serviços de rede [156]. Os parâmetros utilizados nos ataques são mostrados em detalhes na Tabela 12. Esse conjunto de dados está disponível *on-line* [2].

[2] <http://www.uel.br/grupos/orion/datasets.html>



Fonte: O próprio autor.

Figura 32 – Topologia do cenário de rede emulada.

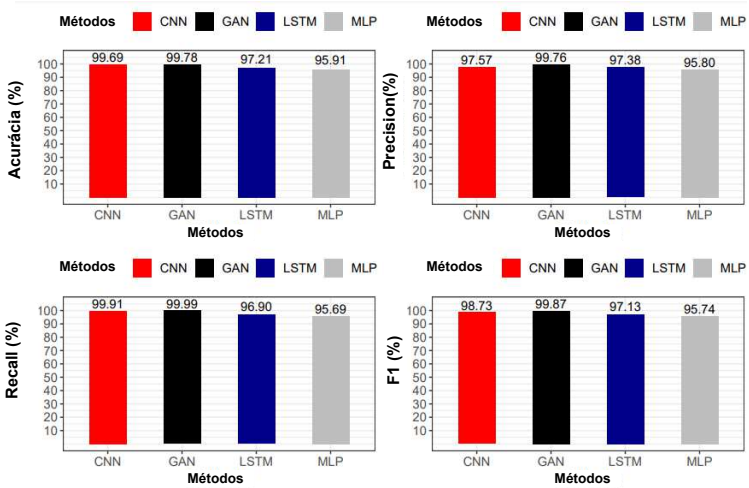
Tabela 12 – UDP DDoS: parâmetros.

Tipo de ataque	**Parâmetros**
DDoS #1	Atacantes: 15 IPs atacantes: 10.0.0.21 - 10.0.0.35 IP Vítima: 10.0.0.92:2000 Horário: 10:10 - 11:20
DDoS #2	Atacantes: 16 IPs atacantes: 10.0.0.45 - 10.0.0.60 IP vítima: 10.0.0.33:8080 Horário: 14:45 - 16:00

Fonte: O próprio autor.

O desempenho do *framework GAN* foi avaliado e comparado ao de outros métodos baseados em redes neurais presentes na literatura, que também foram aplicados para detectar ataques *DDoS* em ambientes *SDN*, sendo eles: a *Convolutional Neural Network* (*CNN*) [13], a *Long Short-Term Memory* [52] e a clássica rede neural *Multilayer Perceptron* (*MLP*) [143]. Esses métodos alcançaram resultados precisos na detecção de ataques em *SDN*.

A Figura 33 mostra os resultados alcançados por cada um dos métodos comparados mediante as métricas avaliadas. Os métodos *GAN*, *CNN*, *LSTM* e *MLP* apresentaram valores superiores a 95%. Para a acurácia, GAN e CNN obtiveram resultados semelhantes,



Fonte: O próprio autor.

Figura 33 – Resultados dos modelos comparados no cenário de rede emulado e o *framework GAN*.

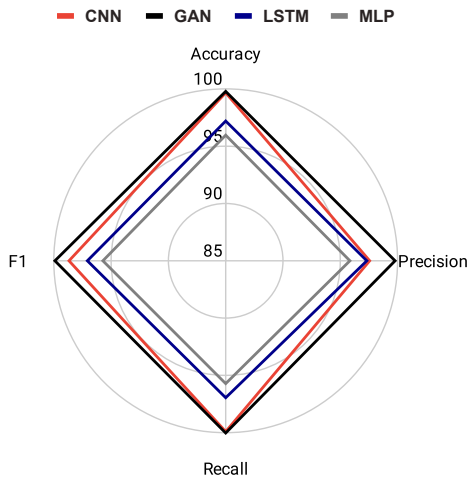
99,78% e 99,69%, respectivamente. A *LSTM* atingiu uma taxa de precisão de 97,21% e a MLP atingiu 95,91%, o que significa que esses dois últimos métodos foram menos precisos na classificação dos intervalos de análise.

A próxima métrica avaliada foi a taxa de *precision*. O *framework GAN* alcançou uma taxa de precisão de 99,76%, uma melhoria de 2,19%, 2,38% e 4,26% superior a *CNN*, *LSTM* e *MLP*, respectivamente. A seguir, foi avaliada a métrica *recall*. O *framework GAN* também atingiu resultados superiores, com uma taxa de 99,99% para essa métrica, seguido dos métodos *CNN*, *LSTM* e *MLP*, atingindo taxas de 99,91%, 96,90% e 95,69%, respectivamente.

Por fim, foi avaliada a robustez de todos os métodos comparados usando a métrica *F1 score*. O *framework GAN* alcançou o melhor resultado, com índice de 99,87%, e os demais foram *CNN*, *LSTM* e *MLP*, atingindo índices de 98,73%, 97,13% e 95,74%, respectivamente.

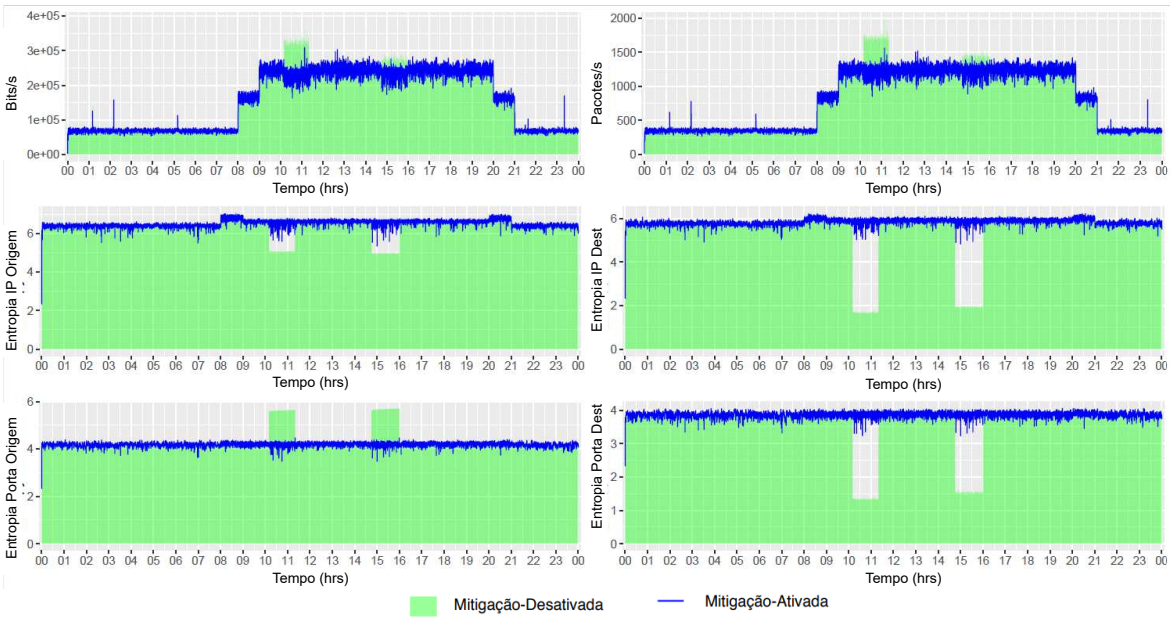
De acordo com os resultados obtidos para todas as métricas avaliadas, o *framework GAN* proposto apresentou um desempenho adequado na detecção de ataques *DDoS*. A Figura 34 mostra um gráfico de radar que resume todas as métricas de desempenho dos métodos comparados.

Os intervalos de análise detectados como ataques pelo módulo de detecção são relatados ao módulo de mitigação. Nesses intervalos, são aplicadas políticas de mitigação. A Figura 35 apresenta o comportamento dos seis atributos de fluxos analisados antes e depois do processo de mitigação. O gráfico de barra verde e o gráfico de linha azul ilustram o tráfego de rede antes e depois de aplicar a política de mitigação. Um aumento nas taxas de *bits* e pacotes pode ser notado antes de aplicar a mitigação. No entanto, após a mitigação, o comportamento do tráfego da rede tende a voltar à sua normalidade, devido aos descartes de pacotes anômalos.



Fonte: O próprio autor.

Figura 34 – Gráfico de radar com os resultados dos métodos avaliados no cenário emulado.



Fonte: O próprio autor.

Figura 35 – Comportamento dos atributos de fluxos antes e depois do processo de mitigação no cenário emulado.

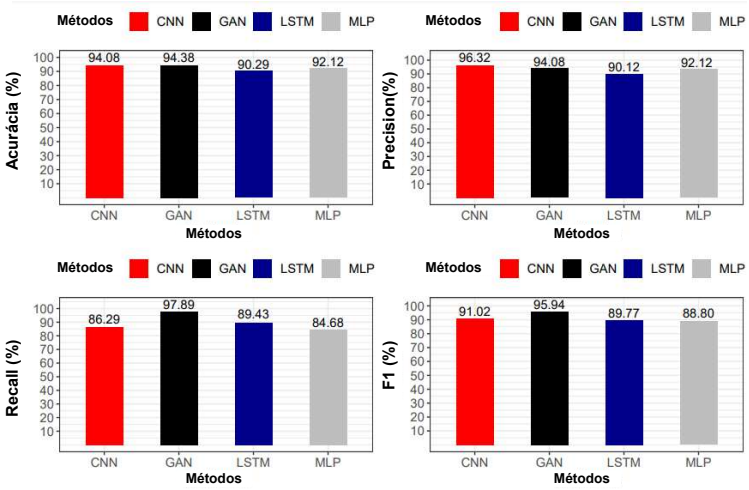
5.3.3 Resultados cenário 2: base de dados CICDDoS 2019

Nesse segundo cenário de teste, foi avaliado o desempenho do sistema proposto para detectar diferentes tipos de ataque *DDoS*. Os ataques avaliados estão presentes no conjunto de dados públicos CICDDoS 2019, que contém 28 perfis normais e tipos recentes de ataques de *DDoS*. O conjunto de dados foi organizado por dia, um para treinamento e outro para teste. O conjunto de dados de treinamento compreende 12 ataques *DDoS* diferentes, como *NTP*, *DNS*, *LDAP*, *MSSQL*, *NetBIOS*, *SNMP*, *SSDP*, *UDP*, *UDP-Lag*, *WebDDoS* (*ARME*), *SYN* e *TFTP*. O dia de teste contém 6 tipos de ataque DDoS, com

NetBIOS, *LDAP*, *MSSQL*, *UDP*, *UDP-Lag* e *SYN*.

As métricas utilizadas para avaliar o desempenho do sistema proposto foram as mesmas aplicadas no cenário anterior: acurácia, *precision*, *recall* e *F1 score*. Os resultados obtidos foram comparados com os métodos *CNN* [13], *LSTM* [52] e *MLP* [143]. A Figura 36 ilustra os resultados das métricas obtidas para cada um deles.

Como pode ser observado na Figura 36, os métodos *GAN* e *CNN* alcançaram resultados semelhantes para a métrica acurácia, com taxas em torno de 94% (*GAN* atingiu 0,3% acima da *CNN*). Os outros dois métodos, *LSTM* e *MLP*, atingiram taxas de Acurácia de 90,29% e 92,12%, respectivamente.



Fonte: O próprio autor.

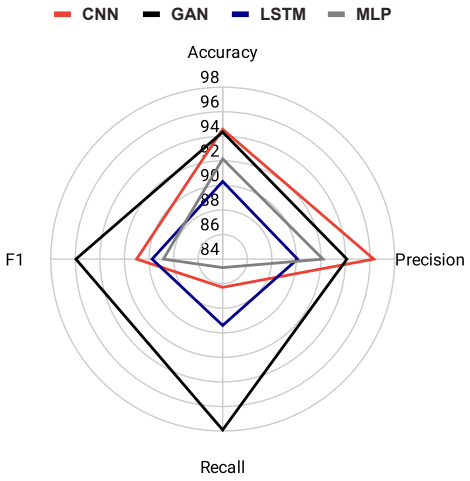
Figura 36 – Resultados dos modelos comparados e o *framework GAN* no conjunto de dados CICDDoS 2019.

Com relação à métrica *precision*, o método CNN se saiu relativamente melhor do que os outros métodos comparados, atingindo uma taxa de 96,32%, seguido por *GAN*, *MLP* e *LSTM*, com taxas de 94,08%, 92,12% e 90,12% , respectivamente. Para a métrica *reacall*, o *framework* GAN obteve desempenho superior, alcançando uma taxa de 97,89%, uma melhoria de taxa de 8,46%, 11,6% e 13,21% maior que *LSTM*, *CNN* e *MLP*, respectivamente.

Conforme mencionado no cenário anterior, também foi utilizada a métrica *F1 score* para determinar a precisão e a robustez dos métodos na detecção de diferentes tipos de ataque *DDoS*. Com relação a essa métrica, fica claro que o *framework GAN* apresentou resultados melhores do que os outros métodos, com uma taxa de 95,94%. Já *CNN*, *LSTM*, *MLP* obtiveram taxas semelhantes, com 91,02%, 89,77% e 88,80%, respectivamente.

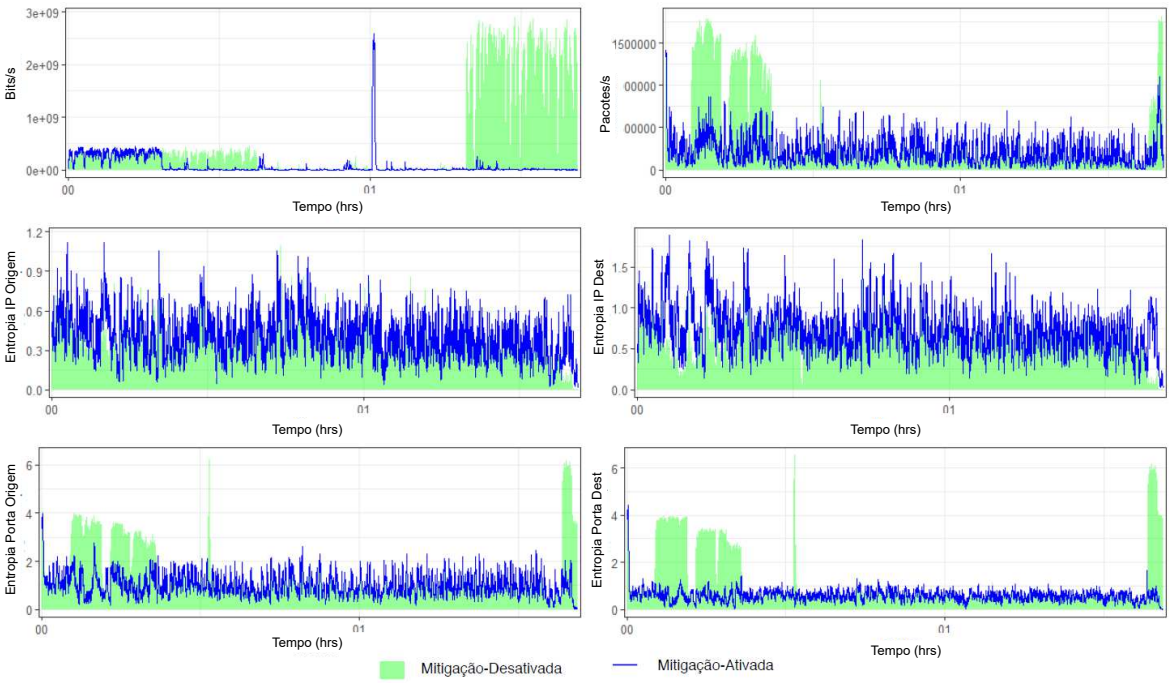
Os resultados das métricas estão resumidos no gráfico de radar ilustrado na Figura 37. Os métodos comparados nesse cenário obtiveram resultados significativamente inferiores em relação ao primeiro cenário. O principal motivo dessa diferença é o número de ataques avaliados, dificultando o processo de detecção. No entanto, o *framework GAN* obteve resultados superiores devido ao seu processo de treinamento, que utilizou uma abordagem

de *adversarial training*, tornando-o menos sensível a diferentes ataques. Como nos resultados obtidos no primeiro cenário, o *framework* GAN também alcançou, em média, os melhores resultados. O sistema proposto fornece uma abordagem eficiente, capaz de detectar diferentes tipos de ataque de *DDoS*.



Fonte: O próprio autor.

Figura 37 – Gráfico de radar com os resultados dos métodos avaliados no conjunto de dados CICDDoS 2019.



Fonte: O próprio Autor.

Figura 38 – Comportamento dos atributos de fluxos antes e depois do processo de mitigação no conjunto de dados CICDDoS 2019.

Nesse cenário, foi avaliado o desempenho da mitigação contra os diversos ataques de *DDoS*. O módulo de detecção de anomalias acionou os intervalos anômalos para o mó-

dulo de mitigação, aplicando automaticamente a política contra os ataques. A Figura 38 mostra o comportamento do tráfego de rede para cada atributo de fluxo. O tráfego antes da mitigação é representado pela área verde e o tráfego após a aplicação da política de mitigação é simbolizado pela linha azul. Como pode ser visto na Figura 38, as variações das características do fluxo foram inibidas após o acionamento do módulo de mitigação, que, por meio dessa análise, conseguiu mitigar os ataques *DDoS* com sucesso e, consequentemente, manter o tráfego normal.

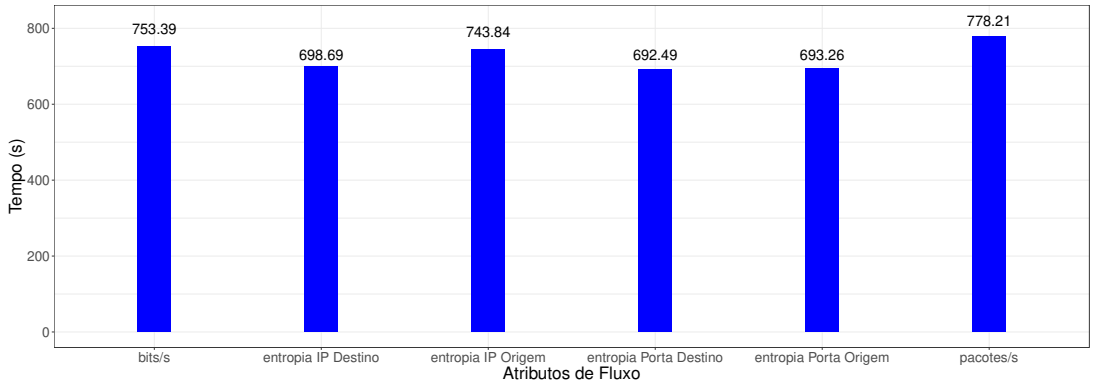
5.4 Análise comparativa: LSTM-FUZZY e o *framework* GAN

Nessa seção foi avaliado e comparado o desempenho do método LSTM-FUZZY e do *framework GAN*, aplicados no módulo de caracterização e detecção. O objetivo dessa avaliação foi identificar o tempo necessário que cada um dos métodos levou na fase de treinamento e na fase de detecção, ou seja, na fase de operação, uma vez que o tempo de processamento é uma característica fundamental para o sistema. Além disso, foram analisados e comparados os desempenhos dos métodos com o objetivo de identificar qual deles apresenta maior robustez com relação às métricas avaliadas, sendo elas *precision*, *recall* e *F1 score*. O processo de avaliação foi utilizando o conjunto de dados CICDDoS 2019, no qual foi aplicado nos cenários anteriores.

5.4.1 Análise de Tempo de Treinamento e Teste

O método LSTM-FUZZY é composto de seis redes LSTM, em que cada uma é responsável pela caracterização de um atributo de fluxo, sendo eles: *bits*/s, pacotes/s, entropia *IP* de origem, entropia *IP* de destino, entropia de porta de origem e entropia de porta de destino. As redes foram treinadas por 200 épocas. Durante a fase de treinamento, foram coletados os dados de tempo de processamento que cada rede. O treinamento da rede responsável pela caracterização do atributo de fluxo de *bits*/s foi de 753,39 segundos, para caracterização de pacotes foi 778,21 segundos, para os atributos de entropia *IP* de origem, entropia *IP* de destino, entropia de porta de origem e entropia de porta de destino foram 743,84, 698,69, 693,26 e 692,49 segundos, respectivamente. O gráfico da Figura 39 ilustra o tempo de treinamento de cada uma das seis LSTM utilizadas. É possível notar que o tempo máximo foi de 778,21 segundos para a dimensão de pacotes, considerando que as redes são independentes e podem ser treinadas em paralelo, portanto esse seria o tempo máximo necessário para o treinamento.

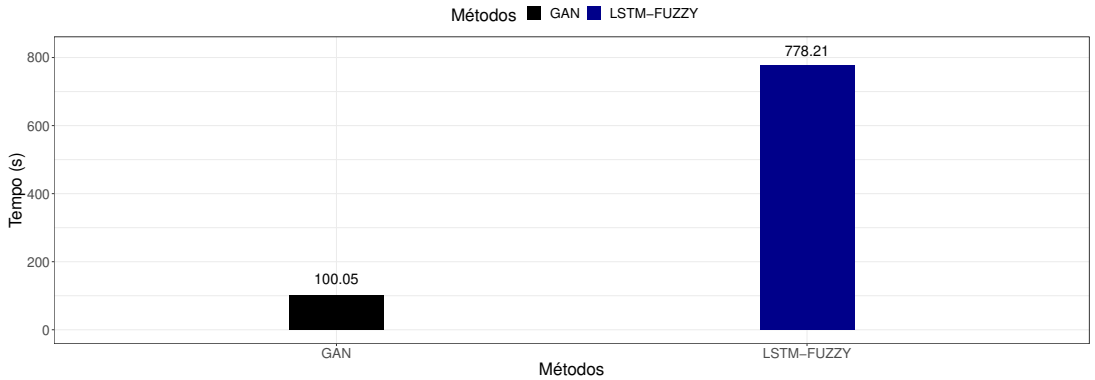
O treinamento do *framework GAN* consiste em treinar simultaneamente duas redes (Gerador e Discriminador) de forma adversária, apresentando as amostras do conjunto de treinamento por e épocas e ajustando os pesos sinápticos até a convergência. No treina-



Fonte: O próprio autor.

Figura 39 – Tempo de treinamento do método LSTM-FUZZY.

mento do *framework* GAN, foram utilizadas 300 épocas de treinamento e o tempo total foi 100,05 segundos. Comparando com o tempo do método LSTM-FUZZY, o *framework* textitGAN leva aproximadamente 7.7x menos tempo em seu treinamento. O tempo maior de treinamento do método LSTM-FUZZY está relacionado com à complexidade arquitetural da *LSTM* utilizada, que apresenta 10.451 mil pesos sinápticos para serem atualizados, enquanto a rede *GAN* tem apenas 541. A Figura 40 mostra o tempo de treinamento dos métodos comparados.



Fonte: O próprio autor.

Figura 40 – Comparação entre os tempos de treinamento do *framework GAN* e o método LSTM-FUZZY.

Os dados de teste extraídos do conjunto da base CICDDoS 2019 são compostos por 7193 amostras. O método LSTM-FUZZY levou 10,58 segundos para caracterizar os seis atributos de fluxos, operando em paralelo, e realizar a atividade de detecção das amostras na fase de teste. Nesse mesmo cenário, o *framework GAN* levou 6,33 segundos para classificar as 7193 amostras do tráfego. Analisando o tempo de resposta para classificar uma amostra do tráfego, o método LSTM-FUZZY leva em média 1.4 milissegundos e o *framework* GAN, 0.88 milissegundos. Comparando o tempo desses métodos, o *framework GAN* pode processar o tráfego em um tempo 1,6x menor em relação ao método LSTM-FUZZY. O tempo de processamento permite que ambos os métodos possam ser usados

em sistemas de detecção de intrusão que atuem em *near real-time*, ou seja, são capazes de operar em cenários nos quais o tráfego de rede é analisado a cada 1 segundo.

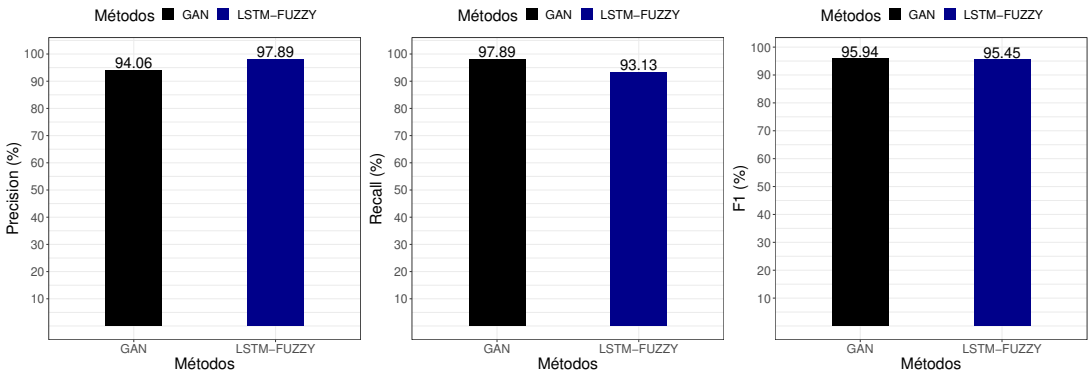
5.4.2 Comparação de desempenho para detecção

Para comparar os resultados do método LSTM-FUZZY e do *framework GAN* na fase de detecção, foram utilizadas as métricas *precision*, *recall* e *F1 score*. Essas métricas permitem analisar o desempenho dos métodos levando em consideração as taxas de falsos-positivos e as de falsos-negativos e o equilíbrio entre elas, indicadores importantes a serem considerados na construção de um sistema de detecção de intrusão.

No cenário de teste, utilizando a base de dados CICDDoS 2019, para a métrica de *precision*, o método LSTM-FUZZY obteve 97,89% e *framework GAN* 94,06%. Analisando esses valores, é possível notar que o método LSTM-FUZZY apresentou um maior equilíbrio entre as taxas de verdadeiro-positivo e falso-positivo, o que indica que o método é menos sensível para classificar amostras normais como anômalas.

Em relação à métrica *recall*, o *framework GAN* se saiu relativamente melhor do que o LSTM-FUZZY, atingindo taxas de 97,89% e 93,13%, respectivamente. Esse resultado indica que o *framework GAN* é menos sensível a falsos-negativos, ou seja, em classificar amostras anômalas como normais. No cenário de detecção de intrusão, os falsos-negativos são mais prejudiciais que os falsos-positivos, pois um ataque não detectado pode onerar o sistema de rede em sua totalidade, tornando-o indisponível e gerando prejuízos aos usuários legítimos que fazem uso do serviço afetado.

Por fim, foi avaliada a robustez dos métodos comparados usando a métrica *F1 score*. O *framework GAN* alcançou um resultado ligeiramente superior, com índice de 95.94%, enquanto método LSTM-FUZZY atingiu 95,45%. De acordo com os resultados obtidos das métricas avaliadas, o *framework GAN* apresentou um desempenho adequado na detecção de ataques *DDoS*, principalmente por ter menor sensibilidade a falsos-negativos. A Figura 41 sumariza os resultados obtidos com as métricas avaliadas dos métodos comparados.



Fonte: O próprio autor.

Figura 41 – Comparação entre os resultados do *framework* GAN e o método LSTM-FUZZY.

6 Conclusões

As redes *SDN* introduziram recursos de programação e centralização da lógica de controle, facilitando as atividades de gerenciamento e flexibilizando a configuração dos ativos de rede por meio de uma interface de programação bem definida. Apesar da introdução desses recursos, assim como nas redes tradicionais, esse ambiente também está suscetível a vulnerabilidades relacionadas à segurança, pois o controle centralizado da rede se torna um alvo ideal para ataques de *DDoS*. Além disso, a quantidade de ataques aumentou em número e na sofisticação em que são executados pelos agentes maliciosos. Dessa maneira, foi apresentada nesta tese uma solução arquitetural modular para a detecção e a mitigação de ataques com base em anomalias e redes neurais profundas em redes *SDN*. Com essa motivação, foram desenvolvidos dois sistemas:

- Sistema 1: *Long Short-Term Memory and Fuzzy Logic for Anomaly Detection and Mitigation in Software-Defined Network Environment*.
- Sistema 2: *Adversarial Deep Learning approach detection and defense against DDoS attacks in SDN environments*.

A seguir, estão descritas as principais contribuições deste trabalho:

- Solução de sistema de defesa modular aplicado em ambientes *SDN*, incluindo os módulos de coleta e processamento de fluxos *IP*, caraterização do tráfego de rede e a detecção e mitigação de eventos anômalo.
- O emprego do modelo de rede neural profunda *Long Short-Term Memory* (*LSTM*) para a caracterização do tráfego de rede.
- Proposta e implementação de um mecanismo de defesa para a detecção de anomalias utilizando o sistema de Inferência Fuzzy.
- Detecção e defesa contra ataques de *DDoS*, aplicando o *framework Generative Adversarial Networks*, que fornece uma taxa de detecção mais precisa e menos sensível a exemplos adversários.
- Coleta e análise do tráfego de rede a cada segundo, permitindo que o sistema de detecção de anomalias atue em tempo quase real (*near real-time*), visto que, quanto menor esse tempo, menor será o tempo de resposta do sistema na detecção das ameaças recebidas.
- Detecção de diferentes tipos de ataque de *DDoS*, por exemplo, *NTP*, *DNS*, *LDAP*, *MSSQL*, *NetBIOS*, *SNMP*, *SSDP*, *UDP*, *UDP-Lag*, *WebDDoS* (*ARME*), *SYN* e *TFTP*.

- Comparação da eficiência das soluções propostas com diferentes métodos de aprendizagem profunda presentes na literatura para a detecção de ataques de *DDoS* em redes *SDN*.

O sistema 1 é composto de quatro módulos, em que suas atividades são realizadas de maneira automatizada para facilitar o monitoramento, a detecção e a mitigação de ataques. No segundo módulo, responsável pela caracterização do tráfego e detecção de anomalias, foi desenvolvida uma nova abordagem semi-supervisionada para prever o comportamento normal de operação da rede, aplicando uma arquitetura de rede neural recorrente profunda, a *Long Short-Term Memory* (*LSTM*). Em conjunto com a caracterização, foi proposto um mecanismo para o reconhecimento de ataques, aplicando a desigualdade de *Bienaymé-Chebyshev* acompanhada da lógica Fuzzy.

Para validar o sistema desenvolvido, foram empregados dois cenários com características distintas. No primeiro foram, aplicados dados *SDN* emulados, utilizando o emulador Mininet e o controlador Floodlight, contendo períodos de ataques de *DDoS* e *Portscan*. No segundo, foi utilizada uma base de dados pública chamada *CICDDoS* 2019. Essa base é composta de 12 tipos diferentes de ataques de *DDoS*. Para testar o módulo de detecção, foi comparado o método LSTM-FUZZY com os outros métodos presentes na literatura, a *SVM*, o *kNN*, a *MLP*, a *LSTM-2* e o *PSO-DS*. A comparação de desempenho entre o método proposto e os demais foi realizada em ambos os cenários. De acordo com os resultados apresentados, o método proposto, LSTM-FUZZY, mostrou um desempenho superior em relação aos demais, obtendo baixas taxas de falsos-positivos e altas taxas *precision*, *recall* e *AUC*.

Nos cenários avaliados, foram aplicadas políticas de mitigação, conforme o tipo de ataque identificado pelo módulo de detecção. Nesse módulo foram identificados os fluxos suspeitos com base na análise dos endereços *IP* e portas que compõem o intervalo anômalo. Os fluxos identificados como suspeitos foram descartados. Por meio do teste de McNemar e da taxa de pacotes anômalos descartados, comprovou-se que o módulo minimizou os efeitos dos ataques.

A vantagem da *LSTM* em aprender e extrair padrões de curto e longo prazo permitiu a sua aplicação para prever o comportamento normal do tráfego de rede. O modelo produziu previsões adequadas próximas ao comportamento real do tráfego e foi possível aplicá-las na fase de detecção. As características da lógica Fuzzy permitiram a detecção de anomalias do modo não supervisionado, permitindo que o sistema não necessite de dados rotulados. O emprego dessa técnica facilita a operação do sistema e descarta a necessidade de utilizar bases de dados rotulados, as quais exigem muito trabalho e podem estar repletas de erros humanos. Além disso, a Lógica Fuzzy atuou na detecção de diferentes ataques de *DDoS* com uma taxa de falsos-positivos baixa, permitindo que o sistema atue no ambiente *SDN* atual com uma alta acurácia na detecção e com baixos falsos alarmes.

Os resultados obtidos demonstram que os módulos que compõem o sistema proposto foram capazes de executar as tarefas designadas a cada um deles. A execução das atividades realizadas pelo sistema é autônoma, isto é, os processos de monitoramento, identificação de eventos adversos e tomada de contramedidas são realizados sem a necessidade de interferência humana. O monitoramento e o gerenciamento da rede são atividades complexas, por isso a aplicação de um sistema autônomo auxilia nas tarefas designadas ao administrador para manter e garantir o funcionamento da rede em sua totalidade. Portanto, o sistema desenvolvido neste trabalho pode ser aplicado para facilitar os procedimentos de gerenciamento e garantir a disponibilidade dos serviços oferecidos.

A arquitetura modular do sistema permite a manutenção e a adaptação de outras técnicas para a caracterização do tráfego e a detecção e mitigação de anomalias em ambientes *SDN*. Essa característica permite ao sistema adaptar-se conforme a dinâmica da rede se altera e surgem novas demandas de segurança. Desse modo, os trabalhos futuros podem explorar outras vulnerabilidades e incorporar políticas de mitigação para suprir as novas demandas que surgirem nos ambientes de redes *SDN*.

Com os resultados extraídos no desenvolvimento do sistema 1, foi publicado na revista *IEEE ACCESS* (Qualis A1; Fator de Impacto 3.9) o artigo intitulado *"Long Short-Term Memory and Fuzzy Logic for Anomaly Detection and Mitigation in Software-Defined Network Environment"* [157], que se encontra no Apêndice A.

No sistema 2, foi proposto e desenvolvido um mecanismo de detecção e defesa contra ataques de *DDoS* recentes em ambientes *SDN*. Os sistema é composto de quatro módulos integrados que fornecem ferramentas para coletar, processar, detectar e mitigar ataques. Destaca-se o emprego do *framework Generative Adversarial Network*, aplicando treinamento adversário para tornar o sistema robusto e menos sensível a exemplos adversários.

Os resultados do sistema proposto foram comparados com diferentes métodos, presentes na literatura, de redes neurais para a detecção de *DDoS* em *SDN*, como *CNN*, *LSTM* e *MLP*. Durante a etapa de teste, os métodos foram submetidos a dois cenários. O primeiro emulou um ambiente *SDN* real com muitos *hosts* e altas taxas de transmissão, usando o emulador *Mininet* e o controlador *Floodlight*. No segundo cenário, foi avaliado o sistema por meio do conjunto de dados público recente *CICDDoS 2019*, que contém os 12 tipos mais comuns e recentes de ataque de *DDoS*.

No primeiro cenário, foi avaliado o sistema proposto para a detecção de ataques *UDP flood* em um ambiente de rede *SDN* com altas taxas de transmissão. O tráfego de rede foi analisado quase em tempo real, em intervalos de um segundo. Os resultados obtidos nesse ambiente de teste pelo sistema proposto foram superiores aos dos outros métodos comparados. De acordo com as métricas avaliadas, o sistema foi capaz de detectar a maioria dos ataques *DDoS* realizados e se defender deles.

No segundo cenário, foi avaliado o desempenho do sistema para a detecção de ataques *DDoS* contra diferentes aplicações e os métodos comparados foram menos precisos na

detecção desses ataques. Por outro lado, foi otimizado o sistema usando o *framework GAN* por meio de treinamento adversário, que forneceu exemplos adversários para aprimorar sua capacidade de defesa contra eles, tornando o sistema mais preciso na identificação dos diferentes ataques *DDoS*. Em ambos os cenários, o sistema proposto obteve resultados superiores aos demais métodos comparados, permitindo sua aplicação contra diversos ataques *DDoS* em ambientes *SDN*.

O *framework* GAN mostrou potencial no uso em ambientes suscetíveis a diferentes ameaças devido ao seu treinamento adversário, que torna o sistema menos sensível a ataques adversários. Além disso, os sistema melhorou os indicadores de desempenho avaliados neste trabalho e se mostrou eficiente em detectar novos ataques de *DDoS* em diferentes aplicações.

Devido à modularidade do sistema, trabalhos futuros podem explorar o impacto de outras arquiteturas de redes neurais profundas (*Gated-Recurrent Unit*, *Stacked Auto-Encoder*, *Convolutional Neural Network*) no Gerador e no Discriminador para aumentar ainda mais o poder de generalização do sistema proposto. Outra oportunidade futura é explorar o processo de detectar e classificar outros ataques de rede em uma abordagem de classificação multiclasse.

Com os resultados extraídos do desenvolvimento do sistema 2, foi publicado na revista *Future Generation Computer Systems* (Qualis A1; Fator de Impacto 7.5) o artigo intitulado *"Adversarial Deep Learning approach detection and defense against DDoS attacks in SDN environments"* [158], o qual se encontra no Apêndice B.

Referências

[1] CUI, J.; WANG, M.; LUO, Y.; ZHONG, H. DDoS detection and defense mechanism based on cognitive-inspired computing in SDN. *Future Generation Computer Systems*, Elsevier B.V., v. 97, p. 275–283, 2019. ISSN 0167-739X. DOI:10.1016/j.future.2019.02.037.

[2] MOUSTAFA, N.; HU, J.; SLAY, J. A holistic review of network anomaly detection systems: A comprehensive survey. *Journal of Network and Computer Applications*, v. 128, p. 33 – 55, 2019. ISSN 1084-8045. DOI:10.1016/j.jnca.2018.12.006.

[3] REGO, A.; GARCIA, L.; SENDRA, S.; LLORET, J. Software defined network-based control system for an efficient traffic management for emergency situations in smart cities. *Future Generation Computer Systems*, v. 88, p. 243 – 253, 2018. ISSN 0167-739X. DOI:10.1016/j.future.2018.05.054.

[4] BARAKABITZE, A. A.; AHMAD, A.; MIJUMBI, R.; HINES, A. 5g network slicing using sdn and nfv: A survey of taxonomy, architectures and future challenges. *Computer Networks*, v. 167, p. 106984, 2020. ISSN 1389-1286. DOI:10.1016/j.comnet.2019.1069.

[5] KREUTZ, D.; RAMOS, F. M. V.; VERISSIMO, P. E.; ROTHENBERG, C. E.; AZODOLMOLKY, S.; UHLIG, S. Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*, IEEE, v. 103, n. 1, p. 14–76, jan 2015. ISSN 0018-9219. DOI:10.1109/JPROC.2014.2371999.

[6] 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. ISSN 2169-3536. DOI:10.1109/ACCESS.2020.2997939.

[7] 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>.

[8] 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>.

[9] JONKER, M.; KING, A.; KRUPP, J.; ROSSOW, C.; SPEROTTO, A.; DAINOTTI, A. Millions of targets under attack: A macroscopic characterization of the dos ecosystem. In: *Proceedings of the 2017 Internet Measurement Conference*. New York, NY, USA: ACM, 2017. (IMC '17), p. 100–113. ISBN 978-1-4503-5118-8. DOI:10.1145/3131365.3131383.

[10] AWS. Threat landscape report – q1 2020. p. 1–9, 2020. Disponível em: <https://aws-shield-tlr.s3.amazonaws.com/2020-Q1_AWS_Shield_TLR.pdf>.

[11] YOON, C.; PARK, T.; LEE, S.; KANG, H.; SHIN, S.; ZHANG, Z. Enabling security functions with SDN: A feasibility study. *Computer Networks*, Elsevier Ltd., v. 85, n. 2015, p. 19–35, 2015. ISSN 13891286. DOI:10.1016/j.comnet.2015.05.005.

[12] KALKAN, K.; GUR, G.; ALAGOZ, F. Defense Mechanisms against DDoS Attacks in SDN Environment. *IEEE Communications Magazine*, v. 55, n. 9, p. 175–179, 2017. ISSN 01636804. DOI:10.1109/MCOM.2017.1600970.

[13] 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. DOI:10.1016/j.compeleceng.2020.106738.

[14] Xin, Y.; Kong, L.; Liu, Z.; Chen, Y.; Li, Y.; Zhu, H.; Gao, M.; Hou, H.; Wang, C. Machine learning and deep learning methods for cybersecurity. *IEEE Access*, v. 6, p. 35365–35381, 2018. ISSN 2169-3536. DOI:10.1109/ACCESS.2018.2836950.

[15] FERNANDES JR., G.; RODRIGUES, J. J.; CARVALHO, L. F.; AL-MUHTADI, J. F.; PROENCA., M. L. A comprehensive survey on network anomaly detection. *Telecommun. Syst.*, v. 70, n. 3, p. 447–489, mar. 2019. ISSN 1018-4864.

[16] MAHDAVIFAR, S.; GHORBANI, A. A. Application of deep learning to cybersecurity: A survey. *Neurocomputing*, 2019. ISSN 18728286. DOI:10.1016/j.neucom.2019.02.056.

[17] Mishra, P.; Varadharajan, V.; Tupakula, U.; Pilli, E. S. A detailed investigation and analysis of using machine learning techniques for intrusion detection. *IEEE Communications Surveys Tutorials*, v. 21, n. 1, p. 686–728, Firstquarter 2019. DOI:10.1109/COMST.2018.2847722.

[18] Khan, F. A.; Gumaei, A.; Derhab, A.; Hussain, A. A novel two-stage deep learning model for efficient network intrusion detection. *IEEE Access*, v. 7, p. 30373–30385, 2019. ISSN 2169-3536. DOI:10.1109/ACCESS.2019.2899721.

[19] Malaiya, R. K.; Kwon, D.; Suh, S. C.; Kim, H.; Kim, I.; Kim, J. An empirical evaluation of deep learning for network anomaly detection. *IEEE Access*, v. 7, p. 140806–140817, 2019. ISSN 2169-3536. DOI:10.1109/ACCESS.2019.2943249.

[20] Vinayakumar, R.; Alazab, M.; Soman, K. P.; Poornachandran, P.; Al-Nemrat, A.; Venkatraman, S. Deep learning approach for intelligent intrusion detection system. *IEEE Access*, v. 7, p. 41525–41550, 2019. ISSN 2169-3536. DOI:10.1109/ACCESS.2019.2895334.

[21] Haider, S.; Akhunzada, A.; Mustafa, I.; Patel, T. B.; Fernandez, A.; Choo, K. R.; Iqbal, J. A deep cnn ensemble framework for efficient ddos attack detection in software defined networks. *IEEE Access*, v. 8, p. 53972–53983, 2020. DOI:10.1109/ACCESS.2020.2976908.

[22] 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>.

[23] 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>.

[24] MACAS, M.; WU, C.; FUERTES, W. A survey on deep learning for cybersecurity: Progress, challenges, and opportunities. *Computer Networks*, v. 212, p. 109032, 2022. ISSN 1389-1286. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1389128622001864>.

[25] Yuan, X.; He, P.; Zhu, Q.; Li, X. Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, v. 30, n. 9, p. 2805–2824, Sep. 2019. ISSN 2162-2388.

[26] PAWLICKI, M.; CHORAś, M.; KOZIK, R. Defending network intrusion detection systems against adversarial evasion attacks. *Future Generation Computer Systems*, v. 110, p. 148 – 154, 2020. ISSN 0167-739X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X20303368>.

[27] HUANG, X.; KROENING, D.; RUAN, W.; SHARP, J.; SUN, Y.; THAMO, E.; WU, M.; YI, X. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review*, v. 37, p. 100270, 2020. ISSN 1574-0137. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1574013719302527>.

[28] ZHOU, S.; LIU, C.; YE, D.; ZHU, T.; ZHOU, W.; YU, P. S. Adversarial attacks and defenses in deep learning: From a perspective of cybersecurity. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 55, n. 8, dec 2022. ISSN 0360-0300. Disponível em: <https://doi-org.ez78.periodicos.capes.gov.br/10.1145/3547330>.

[29] Zhang, X.; Zhou, Y.; Pei, S.; Zhuge, J.; Chen, J. Adversarial examples detection for xss attacks based on generative adversarial networks. *IEEE Access*, v. 8, p. 10989–10996, 2020. ISSN 2169-3536.

[30] KUMAR, V.; SINHA, D. Synthetic attack data generation model applying generative adversarial network for intrusion detection. *Computers & Security*, v. 125, p. 103054, 2023. ISSN 0167-4048. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167404822004461>.

[31] LU, S.; WANG, M.; WANG, D.; WEI, X.; XIAO, S.; WANG, Z.; HAN, N.; WANG, L. Black-box attacks against log anomaly detection with adversarial examples. *Information Sciences*, v. 619, p. 249–262, 2023. ISSN 0020-0255. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0020025522012798>.

[32] GOODFELLOW, I. J.; POUGET-ABADIE, J.; MIRZA, M.; XU, B.; WARDE-FARLEY, D.; OZAIR, S.; COURVILLE, A.; BENGIO, Y. Generative adversarial nets. In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. Cambridge, MA, USA: MIT Press, 2014. (NIPS'14), p. 2672–2680.

[33] ELIYAN, L. F.; Di Pietro, R. Dos and ddos attacks in software defined networks: A survey of existing solutions and research challenges. *Future Generation Computer Systems*, v. 122, p. 149–171, 2021. ISSN 0167-739X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167739X21000911>.

[34] KAUR, S.; KUMAR, K.; AGGARWAL, N.; SINGH, G. A comprehensive survey of ddos defense solutions in sdn: Taxonomy, research challenges, and future directions. *Computers & Security*, v. 110, p. 102423, 2021. ISSN 0167-4048. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167404821002479>.

[35] KHORSANDROO, S.; SáNCHEZ, A. G.; TOSUN, A. S.; ARCO, J.; DORIGUZZI-CORIN, R. Hybrid sdn evolution: A comprehensive survey of the state-of-the-art. *Computer Networks*, v. 192, p. 107981, 2021. ISSN 1389-1286. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1389128621001109>.

[36] HAMAMOTO, A.; CARVALHO, L.; SAMPAIO, L.; ABRÃO, T.; JUNIOR, M. P. Network Anomaly Detection System using Genetic Algorithm and Fuzzy Logic. *Expert Systems with Applications*, Elsevier Ltd, v. 92, p. 390–402, 2018. ISSN 09574174. DOI:10.1016/j.eswa.2017.09.013.

[37] De Assis, M.; HAMAMOTO, A.; AO, T. A.; JUNIOR, M. P. A game theoretical based system using holt-winters and genetic algorithm with fuzzy logic for DoS/DDoS mitigation on SDN networks. *IEEE Access*, v. 5, p. 9485–9496, 2017. ISSN 21693536. DOI:10.1109/ACCESS.2017.2702341.

[38] ZERBINI, C. B.; CARVALHO, L. F.; ABRãO, T.; PROENçA, M. L. Wavelet against random forest for anomaly mitigation in software-defined networking. *Applied Soft Computing*, v. 80, p. 138 – 153, 2019. ISSN 1568-4946. DOI:10.1016/j.asoc.2019.02.046.

[39] JMILA, H.; KHEDHER, M. I. Adversarial machine learning for network intrusion detection: A comparative study. *Computer Networks*, Elsevier B.V., v. 214, 9 2022. ISSN 13891286.

[40] KASIM Ömer. A robust dns flood attack detection with a hybrid deeper learning model. *Computers and Electrical Engineering*, Elsevier Ltd, v. 100, 5 2022. ISSN 00457906.

[41] PINGALE, S. V.; SUTAR, S. R. Remora whale optimization-based hybrid deep learning for network intrusion detection using cnn features. *Expert Systems with Applications*, Elsevier Ltd, v. 210, 12 2022. ISSN 09574174.

[42] ZHU, Y.; CUI, L.; DING, Z.; LI, L.; LIU, Y.; HAO, Z. Black box attack and network intrusion detection using machine learning for malicious traffic. *Computers & Security*, v. 123, p. 102922, 2022. ISSN 0167-4048. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167404822003145>.

[43] ALEROUD, A.; ALSMADI, I. Identifying cyber-attacks on software defined networks: An inference-based intrusion detection approach. *Journal of Network and Computer Applications*, Elsevier, v. 80, n. November 2016, p. 152–164, 2017. ISSN 10958592. DOI:10.1016/j.jnca.2016.12.024.

[44] NOOR, M. binti M.; HASSAN, W. H. Current research on internet of things (iot) security: A survey. *Computer Networks*, v. 148, p. 283 – 294, 2019. ISSN 1389-1286. DOI:10.1016/j.comnet.2018.11.025.

[45] SCARANTI, G. F.; CARVALHO, L. F.; BARBON, S.; LLORET, J.; PROENçA, M. L. Unsupervised online anomaly detection in software defined network environments. *Expert Systems with Applications*, v. 191, p. 116225, 2022. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417421015384>.

[46] CARVALHO, L. F.; ABRãO, T.; MENDES, L. de S.; PROENçA, M. L. An ecosystem for anomaly detection and mitigation in software-defined networking. *Expert Systems with Applications*, v. 104, p. 121 – 133, 2018. ISSN 0957-4174. DOI:10.1016/j.eswa.2018.03.027.

[47] SHONE, N.; NGOC, T. N.; PHAI, V. D.; SHI, Q. A Deep Learning Approach to Network Intrusion Detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, IEEE, v. 2, n. 1, p. 41–50, 2018. DOI:10.1109/tetci.2017.2772792.

[48] ROY, P. K.; SINGH, J. P.; BANERJEE, S. Deep learning to filter SMS Spam. *Future Generation Computer Systems*, v. 102, p. 524–533, 2020. ISSN 0167-739X. DOI:10.1016/j.future.2019.09.001.

[49] YUAN, X.; LI, C.; LI, X. DeepDefense : Identifying DDoS Attack via Deep Learning. *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, IEEE, p. 1–8, 2017. DOI:10.1109/SMARTCOMP.2017.7946998.

[50] TANG, T. A.; MHAMDI, L.; MCLERNON, D.; ZAIDI, S. A. R.; GHOGHO, M. Deep learning approach for Network Intrusion Detection in Software Defined Networking. *Proceedings - 2016 International Conference on Wireless Networks and Mobile Communications, WINCOM 2016: Green Communications and Networking*, IEEE, p. 258–263, 2016. DOI:10.1109/WINCOM.2016.7777224.

[51] LI, C.; WU, Y.; YUAN, X.; SUN, Z.; WANG, W.; LI, X.; GONG, L. Detection and defense of DDoS attack–based on deep learning in OpenFlow-based SDN. *International Journal of Communication Systems*, v. 31, n. 5, p. 1–15, 2018. ISSN 10991131. DOI:10.1002/dac.3497.

[52] PRIYADARSHINI, R.; BARIK, R. K. A deep learning based intelligent framework to mitigate ddos attack in fog environment. *Journal of King Saud University - Computer and Information Sciences*, 2019. ISSN 1319-1578. DOI:10.1016/j.jksuci.2019.04.010.

[53] DEY, S. K.; RAHMAN, M. M. Flow based anomaly detection in software defined networking: A deep learning approach with feature selection method. *4th International Conference on Electrical Engineering and Information and Communication Technology, iCEEiCT 2018*, p. 630–635, 2019. DOI:10.1109/CEEICT.2018.8628069.

[54] Yang, K.; Zhang, J.; Xu, Y.; Chao, J. Ddos attacks detection with autoencoder. In: *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*. [S.l.: s.n.], 2020. p. 1–9. DOI:10.1109/NOMS47738.2020.9110372.

[55] LIU, Y.; ZHI, T.; SHEN, M.; WANG, L.; LI, Y.; WAN, M. Software-defined ddos detection with information entropy analysis and optimized deep learning. *Future Generation Computer Systems*, Elsevier B.V., v. 129, p. 99–114, 4 2022. ISSN 0167739X.

[56] LENT, D. M. B.; NOVAES, M. P.; CARVALHO, L. F.; LLORET, J.; RODRIGUES, J. J. P. C.; PROENçA, M. L. A gated recurrent unit deep learning model to detect and mitigate distributed denial of service and portscan attacks. *IEEE Access*, v. 10, p. 73229–73242, 2022.

[57] ALDWEESH, A.; DERHAB, A.; EMAM, A. Z. Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues. *Knowledge-Based Systems*, v. 189, p. 105124, 2020. ISSN 0950-7051. DOI:10.1016/j.knosys.2019.105124.

[58] MALDONADO, J.; RIFF, M. C.; NEVEU, B. A review of recent approaches on wrapper feature selection for intrusion detection. *Expert Systems with Applications*, v. 198, p. 116822, 2022. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417422002780>.

[59] SARAN, N.; KESSWANI, N. A comparative study of supervised machine learning classifiers for intrusion detection in internet of things. *Procedia Computer Science*, v. 218, p. 2049–2057, 2023. ISSN 1877-0509. International Conference on Machine Learning and Data Engineering. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877050923001813>.

[60] Tidjon, L. N.; Frappier, M.; Mammar, A. Intrusion detection systems: A cross-domain overview. *IEEE Communications Surveys Tutorials*, v. 21, n. 4, p. 3639–3681, Fourthquarter 2019. ISSN 1553-877X. DOI:10.1109/COMST.2019.2922584.

[61] MASDARI, M.; KHEZRI, H. A survey and taxonomy of the fuzzy signature-based intrusion detection systems. *Applied Soft Computing*, v. 92, p. 106301, 2020. ISSN 1568-4946. DOI:10.1016/j.asoc.2020.106301.

[62] JAMALIPOUR, A.; MURALI, S. A taxonomy of machine-learning-based intrusion detection systems for the internet of things: A survey. *IEEE Internet of Things Journal*, v. 9, n. 12, p. 9444–9466, 2022.

[63] AHMED, M.; MAHMOOD, A.; ISLAM, M. A survey of anomaly detection techniques in financial domain. *Future Generation Computer Systems*, Elsevier B.V., v. 55, p. 278–288, 2016. ISSN 0167739X. DOI:10.1016/j.future.2015.01.001.

[64] RUFF, L.; KAUFFMANN, J. R.; VANDERMEULEN, R. A.; MONTAVON, G.; SAMEK, W.; KLOFT, M.; DIETTERICH, T. G.; MüLLER, K.-R. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, v. 109, n. 5, p. 756–795, 2021.

[65] BHUYAN, M. H.; BHATTACHARYYA, D. K.; KALITA, J. K. Network anomaly detection: Methods, systems and tools. *IEEE Communications Surveys Tutorials*, v. 16, n. 1, p. 303–336, First 2014. ISSN 1553-877X. DOI:110.1109/SURV.2013.052213.00046.

[66] CHAGANTI, R.; BHUSHAN, B.; RAVI, V. A survey on blockchain solutions in ddos attacks mitigation: Techniques, open challenges and future directions. *Computer Communications*, v. 197, p. 96–112, 2023. ISSN 0140-3664. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0140366422004145>.

[67] HOANG, D.-T.; KANG, H.-J. A survey on deep learning based bearing fault diagnosis. *Neurocomputing*, v. 335, p. 327 – 335, 2019. ISSN 0925-2312. DOI:10.1016/j.neucom.2018.06.078.

[68] ATHANASIADIS, C.; HORTAL, E.; ASTERIADIS, S. Audio–visual domain adaptation using conditional semi-supervised generative adversarial networks. *Neurocomputing*, v. 397, p. 331 – 344, 2020. ISSN 0925-2312. DOI:10.1016/j.neucom.2019.09.106.

[69] SCHONEVELD, L.; OTHMANI, A.; ABDELKAWY, H. Leveraging recent advances in deep learning for audio-visual emotion recognition. *Pattern Recognition Letters*, v. 146, p. 1–7, 2021. ISSN 0167-8655. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167865521000878>.

[70] KANTAMANENI, S.; CHARLES, A.; BABU, T. R. Speech enhancement with noise estimation and filtration using deep learning models. *Theoretical Computer Science*, v. 941, p. 14–28, 2023. ISSN 0304-3975. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0304397522004935>.

[71] RASHEED, I.; HU, F.; ZHANG, L. Deep reinforcement learning approach for autonomous vehicle systems for maintaining security and safety using lstm-gan. *Vehicular Communications*, v. 26, p. 100266, 2020. ISSN 2214-2096. DOI:10.1016/j.vehcom.2020.100266.

[72] BACHUTE, M. R.; SUBHEDAR, J. M. Autonomous driving architectures: Insights of machine learning and deep learning algorithms. *Machine Learning with Applications*, v. 6, p. 100164, 2021. ISSN 2666-8270. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2666827021000827>.

[73] WEN, L.-H.; JO, K.-H. Deep learning-based perception systems for autonomous driving: A comprehensive survey. *Neurocomputing*, v. 489, p. 255–270, 2022. ISSN 0925-2312. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0925231222003113>.

[74] FERRAG, M. A.; MAGLARAS, L.; MOSCHOYIANNIS, S.; JANICKE, H. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, v. 50, p. 102419, 2020. ISSN 2214-2126. DOI:10.1016/j.jisa.2019.102419.

[75] LIU, X.; YU, W.; LIANG, F.; GRIFFITH, D.; GOLMIE, N. On deep reinforcement learning security for industrial internet of things. *Computer Communications*, v. 168, p. 20–32, 2021. ISSN 0140-3664. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0140366420320193>.

[76] SHARMA, H.; KUMAR, N. Deep learning based physical layer security for terrestrial communications in 5g and beyond networks: A survey. *Physical Communication*, v. 57, p. 102002, 2023. ISSN 1874-4907. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1874490723000058>.

[77] WANG, P.; FAN, E.; WANG, P. Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. *Pattern Recognition Letters*, 2020. ISSN 0167-8655. DOI:10.1016/j.patrec.2020.07.042.

[78] ALGAN, G.; ULUSOY, I. Image classification with deep learning in the presence of noisy labels: A survey. *Knowledge-Based Systems*, v. 215, p. 106771, 2021. ISSN 0950-7051. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950705121000344>.

[79] PLAYOUT, C.; DUVAL, R.; BOUCHER, M. C.; CHERIET, F. Focused attention in transformers for interpretable classification of retinal images. *Medical Image Analysis*, v. 82, p. 102608, 2022. ISSN 1361-8415. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1361841522002377>.

[80] PIKULIAK, M.; ŠIMKO, M.; BIELIKOVá, M. Cross-lingual learning for text processing: A survey. *Expert Systems with Applications*, p. 113765, 2020. ISSN 0957-4174. DOI:10.1016/j.eswa.2020.113765.

[81] CHEN, Y.; SHU, H.; XU, W.; YANG, Z.; HONG, Z.; DONG, M. Transformer text recognition with deep learning algorithm. *Computer Communications*, v. 178, p. 153–160, 2021. ISSN 0140-3664. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0140366421001754>.

[82] GOOSSENS, A.; De Smedt, J.; VANTHIENEN, J. Extracting decision model and notation models from text using deep learning techniques. *Expert Systems with Applications*, v. 211, p. 118667, 2023. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417422017043>.

[83] HINTON, G. E.; SALAKHUTDINOV, R. R. Reducing the dimensionality of data with neural networks. *Science*, American Association for the Advancement of Science, v. 313, n. 5786, p. 504–507, 2006. ISSN 0036-8075. DOI:10.1126/science.1127647.

[84] VINCENT, P.; LAROCHELLE, H.; LAJOIE, I.; BENGIO, Y.; MANZAGOL, P.-A. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, JMLR.org, v. 11, p. 3371–3408, dez. 2010. ISSN 1532-4435. DOI:10.5555/1756006.1953039.

[85] ACKLEY, D. H.; HINTON, G. E.; SEJNOWSKI, T. J. A learning algorithm for boltzmann machines. *Cognitive Science*, v. 9, n. 1, p. 147 – 169, 1985. ISSN 0364-0213. DOI:10.1016/S0364-0213(85)80012-4.

[86] HINTON, G. E. A practical guide to training restricted boltzmann machines. In: _____. *Neural Networks: Tricks of the Trade: Second Edition*. [S.l.]: Springer Berlin Heidelberg, 2012. p. 599–619. ISBN 978-3-642-35289-8. DOI:10.1007/978-3-642-35289-8_32.

[87] SALAKHUTDINOV, R.; HINTON, G. Deep boltzmann machines. In: DYK, D. van; WELLING, M. (Ed.). *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, 2009. (Proceedings of Machine Learning Research, v. 5), p. 448–455. Disponível em: <http://proceedings.mlr.press/v5/salakhutdinov09a.html>.

[88] HOPFIELD, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, National Academy of Sciences, v. 79, n. 8, p. 2554–2558, 1982. ISSN 0027-8424. DOI:10.1073/pnas.79.8.2554.

[89] HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Computation*, MIT Press, Cambridge, MA, USA, v. 9, n. 8, p. 1735–1780, nov. 1997. ISSN 0899-7667. DOI:10.1162/neco.1997.9.8.1735.

[90] CHUNG, J.; GÜLÇEHRE, Ç.; CHO, K.; BENGIO, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv e-prints*, abs/1412.3555, 2014. Presented at the Deep Learning workshop at NIPS2014. Disponível em: <https://arxiv.org/abs/1412.3555>.

[91] DENG, L.; YU, D. Deep learning: Methods and applications. *Found. Trends Signal Process.*, Now Publishers Inc., Hanover, MA, USA, v. 7, n. 3–4, p. 197–387, jun. 2014. ISSN 1932-8346. DOI:10.1561/2000000039.

[92] Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, v. 86, n. 11, p. 2278–2324, 1998. DOI:10.1109/5.72679.

[93] CHAKRABORTY, A.; ALAM, M.; DEY, V.; CHATTOPADHYAY, A.; MUKHOPADHYAY, D. A survey on adversarial attacks and defences. *CAAI Transactions on Intelligence Technology*, v. 6, n. 1, p. 25–45, 2021. Disponível em: <https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/cit2.12028>.

[94] ROSENBERG, I.; SHABTAI, A.; ELOVICI, Y.; ROKACH, L. Adversarial machine learning attacks and defense methods in the cyber security domain. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 54, n. 5, may 2021. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3453158>.

[95] YERLIKAYA, F. A.; BAHTIYAR Şerif. Data poisoning attacks against machine learning algorithms. *Expert Systems with Applications*, v. 208, p. 118101, 2022. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417422012933>.

[96] VERDE, L.; MARULLI, F.; MARRONE, S. Exploring the impact of data poisoning attacks on machine learning model reliability. *Procedia Computer Science*, v. 192, p. 2624–2632, 2021. ISSN 1877-0509. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 25th International Conference KES2021. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877050921017695>.

[97] RATHORE, H.; SASAN, A.; SAHAY, S. K.; SEWAK, M. Defending malware detection models against evasion based adversarial attacks. *Pattern Recognition Letters*, v. 164, p. 119–125, 2022. ISSN 0167-8655. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167865522003026>.

[98] DEBICHA, I.; BAUWENS, R.; DEBATTY, T.; DRICOT, J.-M.; KENAZA, T.; MEES, W. Tad: Transfer learning-based multi-adversarial detection of evasion attacks against network intrusion detection systems. *Future Generation*

Computer Systems, v. 138, p. 185–197, 2023. ISSN 0167-739X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167739X22002722>.

[99] KRAVCHIK, M.; DEMETRIO, L.; BIGGIO, B.; SHABTAI, A. Practical evaluation of poisoning attacks on online anomaly detectors in industrial control systems. *Computers & Security*, v. 122, p. 102901, 2022. ISSN 0167-4048. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167404822002942>.

[100]XIE, F.; GAO, Y.; WANG, J.; ZHAO, W. Defending local poisoning attacks in multi-party learning via immune system. *Knowledge-Based Systems*, v. 238, p. 107850, 2022. ISSN 0950-7051. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950705121010364>.

[101]WANG, C.; CHEN, J.; YANG, Y.; MA, X.; LIU, J. Poisoning attacks and countermeasures in intelligent networks: Status quo and prospects. *Digital Communications and Networks*, v. 8, n. 2, p. 225–234, 2022. ISSN 2352-8648. Disponível em: <https://www.sciencedirect.com/science/article/pii/S235286482100050X>.

[102]MARULLI, F.; VERDE, L.; CAMPANILE, L. Exploring data and model poisoning attacks to deep learning-based nlp systems. *Procedia Computer Science*, v. 192, p. 3570–3579, 2021. ISSN 1877-0509. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 25th International Conference KES2021. Disponível em: <https://www.sciencedirect.com/science/article/pii/S187705092101869X>.

[103]CHAN, P. P.; ZHENG, J.; LIU, H.; TSANG, E.; YEUNG, D. S. Robustness analysis of classical and fuzzy decision trees under adversarial evasion attack. *Applied Soft Computing*, v. 107, p. 107311, 2021. ISSN 1568-4946. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1568494621002349>.

[104]FANG, Z.; WANG, J.; GENG, J.; ZHOU, Y.; KAN, X. A3cmal: Generating adversarial samples to force targeted misclassification by reinforcement learning. *Applied Soft Computing*, v. 109, p. 107505, 2021. ISSN 1568-4946. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1568494621004282>.

[105]SHEN, Z.; YANG, H.; ZHANG, S. Neural network approximation: Three hidden layers are enough. *Neural Networks*, v. 141, p. 160–173, 2021. ISSN 0893-6080. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0893608021001465>.

[106]KINGMA, D.; BA, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[107]DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, v. 12, p. 2121–2159, 07 2011.

[108]ZEILER, M. D. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012. Disponível em: <http://arxiv.org/abs/1212.5701>.

[109]KINGMA, D.; BA, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[110]DOZAT, T. Incorporating Nesterov Momentum into Adam. In: *Proceedings of the 4th International Conference on Learning Representations*. [S.l.: s.n.]. p. 1–4.

[111]DAUPHIN, Y. N.; VRIES, H. de; CHUNG, J.; BENGIO, Y. Rmsprop and equilibrated adaptive learning rates for non-convex optimization. *CoRR*, abs/1502.04390, 2015. Disponível em: <http://dblp.uni-trier.de/db/journals/corr/corr1502.html#DauphinVCB15>.

[112]CHOLLET, F. et al. *Keras*. 2015. <https://keras.io>.

[113]SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, v. 15, n. 56, p. 1929–1958, 2014. Disponível em: <http://jmlr.org/papers/v15/srivastava14a.html>.

[114]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. DOI:10.1016/j.jnca.2020.102595.

[115]MASOUDI, R.; GHAFFARI, A. Software defined networks: A survey. *Journal of Network and Computer Applications*, v. 67, p. 1–25, 2016. ISSN 1084-8045. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1084804516300297>.

[116]PRIYADARSINI, M.; BERA, P. Software defined networking architecture, traffic management, security, and placement: A survey. *Computer Networks*, v. 192, p. 108047, 2021. ISSN 1389-1286. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1389128621001481>.

[117]YANG, L.; NG, B.; SEAH, W. K.; GROVES, L.; SINGH, D. A survey on network forwarding in software-defined networking. *Journal of Network and Computer Applications*, v. 176, p. 102947, 2021. ISSN 1084-8045. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1084804520304021>.

[118]ONF. "openflow switch specification 1.5.1. 2015. Disponível em: <https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf>.

[119]AITKEN, P.; CLAISE, B.; TRAMMELL, B. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information*. [S.l.], 2013. Disponível em: <https://tools.ietf.org/html/rfc7011>.

[120]PROENCA, M.; COPPELMANS, C.; BOTTOLI, M.; ALBERTI, A.; MENDES, L. S. The hurst parameter for digital signature of network segment. In: _____. *Telecommunications and Networking - ICT 2004: 11th International Conference on Telecommunications, Fortaleza, Brazil, August 1-6, 2004. Proceedings*. [S.l.]: Springer Berlin Heidelberg, 2004. p. 772–781. ISBN 978-3-540-27824-5. DOI:10.1007/978-3-540-27824-5_103.

[121]CARVALHO, L.; ABRAO, T.; MENDES, L.; Proença Jr., M. An ecosystem for anomaly detection and mitigation in software-defined networking. *Expert Systems with Applications*, v. 104, p. 121 – 133, 2018. ISSN 0957-4174. DOI:10.1016/j.eswa.2018.03.027.

[122]Garg, S.; Kaur, K.; Kumar, N.; Kaddoum, G.; Zomaya, A. Y.; Ranjan, R. A hybrid deep learning-based model for anomaly detection in cloud datacenter networks. *IEEE Transactions on Network and Service Management*, v. 16, n. 3, p. 924–935, 2019.

[123]ZHOU, Y.; CHENG, G.; JIANG, S.; DAI, M. Building an efficient intrusion detection system based on feature selection and ensemble classifier. *Computer Networks*, v. 174, p. 107247, 2020. ISSN 1389-1286. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128619314203>.

[124]PENA, E. H. M.; BARBON, S.; RODRIGUES, J. J. P. C.; PROENçA, M. L. Anomaly detection using digital signature of network segment with adaptive arima model and paraconsistent logic. In: *2014 IEEE Symposium on Computers and Communications (ISCC)*. [S.l.: s.n.], 2014. p. 1–6. ISSN 1530-1346.

[125]SHANNON, C. E. A mathematical theory of communication. *The Bell System Technical Journal*, v. 27, n. 3, p. 379–423, July 1948. ISSN 0005-8580. DOI:10.1002/j.1538-7305.1948.tb01338.x.

[126]GIOTIS, K.; ARGYROPOULOS, C.; ANDROULIDAKIS, G.; KALOGERAS, D.; MAGLARIS, V. Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments. *Computer Networks*, v. 62, p. 122 – 136, 2014. ISSN 1389-1286. DOI:10.1016/j.bjp.2013.10.014.

[127]Proenca, M. L.; Zarpelao, B. B.; Mendes, L. S. Anomaly detection for network servers using digital signature of network segment. In: *Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop (AICT/SAPIR/ELETE'05)*. [S.l.: s.n.], 2005. p. 290–295. DOI:10.1109/AICT.2005.26.

[128]JR., M. L. P.; JR., G. F.; CARVALHO, L. F.; ASSIS, M. V. O. de; RODRIGUES, J. J. P. C. Digital signature to help network management using flow analysis. *International Journal of Network Management*, v. 26, n. 2, p. 76–94, 2016. DOI:10.1002/nem.1892.

[129]PENA, E. H.; CARVALHO, L. F.; JR., S. B.; RODRIGUES, J. J.; JR., M. L. P. Anomaly detection using the correlational paraconsistent machine with digital signatures of network segment. *Information Sciences*, v. 420, p. 313 – 328, 2017. ISSN 0020-0255. DOI:10.1016/j.ins.2017.08.074.

[130]De Assis, M. V. O.; Novaes, M. P.; Zerbini, C. B.; Carvalho, L. F.; Abrãao, T.; Proença, M. L. Fast defense system against attacks in software defined networks. *IEEE Access*, v. 6, p. 69620–69639, 2018. DOI:10.1109/ACCESS.2018.2878576.

[131]Munir, M.; Siddiqui, S. A.; Dengel, A.; Ahmed, S. Deepant: A deep learning approach for unsupervised anomaly detection in time series. *IEEE Access*, v. 7, p. 1991–2005, 2019. DOI:10.1109/ACCESS.2018.2886457.

[132]YANG, B.; SUN, S.; LI, J.; LIN, X.; TIAN, Y. Traffic flow prediction using lstm with feature enhancement. *Neurocomputing*, v. 332, p. 320 – 327, 2019. ISSN 0925-2312. DOI:10.1016/j.neucom.2018.12.016.

[133]LAI, G.; CHANG, W.-C.; YANG, Y.; LIU, H. Modeling long- and short-term temporal patterns with deep neural networks. In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. New York, NY, USA: Association for Computing Machinery, 2018. (SIGIR '18), p. 95–104. ISBN 9781450356572. DOI:10.1145/3209978.3210006.

[134]BHATIA, J.; DAVE, R.; BHAYANI, H.; TANWAR, S.; NAYYAR, A. Sdn-based real-time urban traffic analysis in vanet environment. *Computer Communications*, v. 149, p. 162 – 175, 2020. ISSN 0140-3664. DOI:110.1016/j.comcom.2019.10.011.

[135]QING, X.; NIU, Y. Hourly day-ahead solar irradiance prediction using weather forecasts by LSTM. *Energy*, Elsevier Ltd, v. 148, p. 461–468, 2018. ISSN 03605442. DOI:10.1016/j.energy.2018.01.177.

[136]AMIDAN, B.; FERRYMAN, T.; COOLEY, S. Data outlier detection using the chebyshev theorem. *IEEE Aerospace Conference Proceedings*, v. 2005, p. 3–8, 2005. ISSN 1095323X. DOI:10.1109/AERO.2005.1559688.

[137]TAYLOR, C.; ALVES-FOSS, J. An empirical analysis of nate: Network analysis of anomalous traffic events. In: *Proceedings of the 2002 Workshop on New Security Paradigms*. New York, NY, USA: ACM, 2002. (NSPW '02), p. 18–26. ISBN 1-58113-598-X. DOI:10.1145/844102.844106.

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

[139]ROSYARA, U. R.; VROMMAN, D.; DUVEILLER, E. Canopy temperature depression as an indication of correlative measure of spot blotch resistance and heat stress tolerance in spring wheat. *Journal of Plant Pathology*, v. 90, n. 1, p. 103–107, 2008. ISSN 11254653. DOI:10.4454/jpp.v90i1.598.

[140]SINGH, J.; BEHAL, S. Detection and mitigation of ddos attacks in sdn: A comprehensive review, research challenges and future directions. *Computer Science Review*, v. 37, p. 100279, 2020. ISSN 1574-0137. DOI:10.1016/j.cosrev.2020.100279.

[141]KASIM Ömer. An efficient and robust deep learning based network anomaly detection against distributed denial of service attacks. *Computer Networks*, v. 180, p. 107390, 2020. ISSN 1389-1286. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128620304114>.

[142]GROSSE, K.; PAPERNOT, N.; MANOHARAN, P.; BACKES, M.; MCDANIEL, P. Adversarial examples for malware detection. In: FOLEY, S. N.; GOLLMANN, D.; SNEKKENES, E. (Ed.). *Computer Security – ESORICS 2017*. Cham: Springer International Publishing, 2017. p. 62–79. ISBN 978-3-319-66399-9.

[143]WANG, M.; LU, Y.; QIN, J. A dynamic mlp-based ddos attack detection method using feature selection and feedback. *Computers & Security*, v. 88, p. 101645, 2020. ISSN 0167-4048. DOI:10.1016/j.cose.2019.101645.

[144]DIXIT, P.; SILAKARI, S. Deep learning algorithms for cybersecurity applications: A technological and status review. *Computer Science Review*, v. 39, p. 100317, 2021.

ISSN 1574-0137. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1574013720304172>.

[145]ABADI, M. et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Software available from tensorflow.org. Disponível em: <http://tensorflow.org/>.

[146]FAWCETT, T. An introduction to roc analysis. *Pattern Recogn. Lett.*, Elsevier Science Inc., New York, NY, USA, v. 27, n. 8, p. 861–874, jun. 2006. ISSN 0167-8655. DOI:10.1016/j.patrec.2005.10.010.

[147]SUN, X.; YANG, Z. Generalized mcnemar's test for homogeneity of the marginal distributions. In: *SAS Global forum*. [s.n.], 2008. v. 382, p. 1–10. Disponível em: <http://www2.sas.com/proceedings/forum2008/382-2008.pdf>.

[148]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. ISSN 1071-6572. DOI:10.1109/CCST.2019.8888419.

[149]TEAM, M. *Mininet Overview*. 2020. Online Access: 2020-08-27 http://mininet.org/overview/. Disponível em: <http://mininet.org/overview/>.

[150]BIONDI, P.; COMMUNITY the S. *Scapy*. 2020. Online Access: 2020-08-27 http://www.secdev.org/projects/scapy/. Disponível em: <http://www.secdev.org/projects/scapy/>.

[151]FLOODLIGHT, P. *Floodlight*. 2020. Online Access: 2020-08-27 http://www.projectfloodlight.org/. Disponível em: <http://www.projectfloodlight.org/>.

[152]Wang, Y.; Liu, J.; Mišić, J.; Mišić, V. B.; Lv, S.; Chang, X. Assessing optimizer impact on dnn model sensitivity to adversarial examples. *IEEE Access*, v. 7, p. 152766–152776, 2019.

[153]XIAO, P.; QU, W.; QI, H.; LI, Z. Detecting ddos attacks against data center with correlation analysis. *Computer Communications*, v. 67, p. 66 – 74, 2015. ISSN 0140-3664. DOI:10.1016/j.comcom.2015.06.012.

[154]Phan, T. V.; Van Toan, T.; Van Tuyen, D.; Huong, T. T.; Thanh, N. H. Openflowsia: An optimized protection scheme for software-defined networks from flooding attacks. In: *2016 IEEE Sixth International Conference on Communications and Electronics (ICCE)*. [S.l.: s.n.], 2016. p. 13–18. DOI:10.1109/CCE.2016.7562606.

[155]VIEGAS, E.; SANTIN, A.; BESSANI, A.; NEVES, N. Bigflow: Real-time and reliable anomaly-based intrusion detection for high-speed networks. *Future Generation Computer Systems*, v. 93, p. 473 – 485, 2019. ISSN 0167-739X. DOI:10.1016/j.future.2018.09.051.

[156]YUREKTEN, O.; DEMIRCI, M. Sdn-based cyber defense: A survey. *Future Generation Computer Systems*, v. 115, p. 126 – 149, 2021. ISSN 0167-739X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X20303277>.

[157]NOVAES, M. P.; CARVALHO, L. F.; LLORET, J.; PROENçA, M. L. Long short-term memory and fuzzy logic for anomaly detection and mitigation in software-defined network environment. *IEEE Access*, v. 8, p. 83765–83781, 2020.

[158]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>.

Trabalhos publicados pelo autor

Trabalhos publicados:

1. **M. P. Novaes**, L. F. Carvalho, J. Lloret and M. L. Proença, "Long Short-Term Memory and Fuzzy Logic for Anomaly Detection and Mitigation in Software-Defined Network Environment", in IEEE Access, vol. 8, pp. 83765-83781, 2020, doi: 10.1109/ACCESS.2020.2992044 (**Fator de Impacto: 3.9**).

2. **M. P. Novaes**, L. F. Carvalho, J. Lloret and M. L. Proença, "Adversarial deep learning approach detection and defense against DDoS attacks in SDN environments", in Future Gener. Comput. Syst., vol. 125, pp. 156-167, 2021, doi: 10.1016/j.future.2021.06.047 (**Fator de Impacto: 7.5**).

3. 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 (**Fator de Impacto: 3.9**).

Apêndice A: Long Short-Term Memory and Fuzzy Logic for Anomaly Detection and Mitigation in Software-Defined Network Environment

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 31, 2020, accepted April 21, 2020, date of publication May 4, 2020, date of current version May 18, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2992044

Long Short-Term Memory and Fuzzy Logic for Anomaly Detection and Mitigation in Software-Defined Network Environment

MATHEUS P. NOVAES[1], LUIZ F. CARVALHO[2], JAIME LLORET[3], (Senior Member, IEEE), AND MARIO LEMES PROENÇA, Jr.[4]

[1]Electric Engineering Department, State University of Londrina (UEL), Londrina 86057-970, Brazil
[2]Computer Engineering Department, Federal Technology University of Paraná (UTFPR), Apucarana 86812-460, Brazil
[3]Integrated Management Coastal Research Institute, Universitat Politecnica de Valencia, 46022 Valencia, Spain
[4]Computer Science Department, State University of Londrina (UEL), Londrina 86057-970, Brazil

Corresponding author: Mario Lemes Proença, Jr. (proenca@uel.br)

This work was supported in part by the National Council for Scientific and Technological Development (CNPq) of Brazil under Project 310668/2019-0, in part by the SETI/Fundação Araucária due to the concession of scholarships, and in part by the Ministerio de Economía y Competitividad through the Programa Estatal de Fomento de la Investigación Científica y Técnica de Excelencia, Subprograma Estatal de Generación de Conocimiento, under Grant TIN2017-84802-C2-1-P.

ABSTRACT Computer networks become complex and dynamic structures. As a result of this fact, the configuration and the managing of this whole structure is a challenging activity. Software-Defined Networks(SDN) is a new network paradigm that, through an abstraction of network plans, seeks to separate the control plane and data plane, and tends as an objective to overcome the limitations in terms of network infrastructure configuration. As in the traditional network environment, the SDN environment is also liable to security vulnerabilities. This work presents a system of detection and mitigation of Distributed Denial of Service (DDoS) attacks and Portscan attacks in SDN environments (LSTM-FUZZY). The LSTM-FUZZY system presented in this work has three distinct phases: characterization, anomaly detection, and mitigation. The system was tested in two scenarios. In the first scenario, we applied IP flows collected from the SDN Floodlight controllers through emulation on Mininet. On the other hand, in the second scenario, the CICDDoS 2019 dataset was applied. The results gained show that the efficiency of the system to assist in network management, detect and mitigate the occurrence of the attacks.

INDEX TERMS DDoS, deep learning, fuzzy, LSTM, portscan, SDN.

I. INTRODUCTION

Nowadays, the number of applications and services that use the Internet has increased quickly. The network system has become complex structures due to a large number of devices that make them, for example, firewall, intrusion detection system, load balancer, switches, routers, etc. In the traditional network environment, each network asset uses complex protocols, and its configuration differs between makers. With the advent of Cloud Computing and the increase in virtualization technologies, the traditional management architecture of the network is not adequate for these applications, particularly at the current data centers [1].

Despite Software-Defined Networking (SDN) not having been created with a specific objective for virtualization functions of the network, it is an emerging network architecture that projects future networks and that meets the new demands of already existing applications [2], [3]. The main characteristic of SDN architecture is the separation of the control and data plane, which means that the control plane is removed from the network device and centralized on a controller [4], [5]. The centralization of the control plane provides a global view of the network and allows the management of its components through an open and well-defined software interface [6].

Along with the increased demand for web applications and the popularization of new IoT (Internet of Things) devices, issues related to security emerge, for example, attacks [7], [8]. The number of attacks has increased in numbers and in the sophistication of how they are carried out by malicious agents, especially the Distributed Denial of Services (DDoS). The purpose of DDoS is to exhaust a resource, even at the

The associate editor coordinating the review of this manuscript and approving it for publication was Amjad Ali.

server level where the attacker, through many solicitations, tries to deprive some service or at the infrastructure level where the attacker saturates a network link [9], [10].

Although the SDN networks have introduced programming and centralization resources of the control logic, which facilitate its management, these resources are the main security vulnerabilities presented by the network architecture [11], [12]. Due to the SDN network architecture, it is known that the management of network flows is centralized and executed by a controller, which is subject to security threats, for example, DDoS attacks, Portscan, IP spoofing, etc [10], [13]–[16]. During a DDoS attack, the network services are overloaded due to the large number of requests sent by the attackers. The controller is the central point of the SDN network and is vulnerable to attacks. Besides, DDoS attacks are followed by Portscan attacks, where the attacker scans for open ports to perform intrusions. Thus, the SDN network security remains undefined and it is necessary the development of solutions related to detection and mitigation of attacks.

In the last few years, with the increase of security threats and the huge enormous volume of traffic, several approaches have been proposed to detect anomalies [17]–[19]. Network anomaly detection consists of two main approaches: Signature-based and Profile-based. For the first one, Signature-based, a database containing signatures of the most diverse kinds of attacks is needed, and the detection of an anomalous event occurs when there is a ''math'' between the behavior of the network and the known pattern attack. On the other hand, the profile-based approach, based on network history data, a prediction of its usual behavior is made, and an anomaly is detected when the predicted behavior and the real behavior diverge from one another [20]. One of the main advantages of this kind of methodology is the detection of unknown anomalies, for the system does not require learning the behavior of the many existing attacks. Furthermore, the current attacks are dynamic, and new patterns emerge frequently [7], [17].Thus, Signature-based approaches demand that the signature of the attacks are updated each time a new attack emerges, resulting in a drawback for the system.

Generally, the anomaly detection techniques intend to recognize sensitive traffic patterns through sudden changes in the expected traffic volume or unexpected changes in the distribution of specific network traffic characteristics, such as IP addresses and ports. The implementation of Machine Learning algorithms provides solutions for detecting and classifying anomalies [21], [22]. These algorithms have the capacity of learning patterns from a set of data and making predictions based on these data. Usually, the Machine Learning techniques employed in anomaly detection systems are divided into two approaches: Shallow Learning and Deep Learning. Shallow Learning algorithms have some limitations, such as largely depending on attributes used in the process of training, and an intensive analysis is necessary in order to capture the most relevant attributes and statistics of the traffic [23], [24]. Besides, the models often need to be retrained to learn new patterns of network behavior [25], [26]. Recently, the methods based on Deep Learning have been applied in many works related to intrusion detection systems, due to the learning capacity and generalization of employed attributes [27]–[29].

Thus, we present a modular system for anomaly detection and mitigation applied on SDN networks environments. The developed system consists of three modules with well-defined functions. The first module is the characterization one, which employs a Deep Learning algorithm called Long short-term memory (LSTM), an architecture of artificial recurrent neural network (RNN), to predict the normal behavior of the network traffic. The second module is responsible for detecting anomalous events, in which the Bienaymé-Chebyshev inequality is applied to generate normality threshold dynamically, and with that, the Fuzzy logic is applied to identify the occurrence of an anomaly in a certain moment of the analysis. The third module of the system is responsible for the mitigation of detected anomalies, intending to minimalize the damages caused by an attacker.

The main contributions of this work are:

- Network traffic characterization employing a Deep Learning LSTM mechanism;
- DDoS and Portscan attacks detection using a Fuzzy inference system;
- Analysis of the network traffic performed in near real-time;
- Test with two datasets containing many kinds of DDoS attacks;
- Comparison between the developed system and other methods present in the literature.

The rest of this work is organized as the following: Section II presents the related works; Section III presents the fundamentals used in the development of the system; Section IV we discuss the system performance results. Ultimately, on Section V the conclusions obtained with the development of this paper is presented.

II. RELATED WORKS

Nowadays, SDN networks are used broadly, however they present many problems related to security [4], [6], [13], [30]. Thus, the SDN network security remains indefinite, and solutions related to detection and mitigation of attacks have been developed [14].

According to AlEroud and Alsmadi [31], when the packet forwarding logic is centralized and allocated in the controller, the malicious agents explore vulnerabilities on the controller, on links of communication between controller and forwarding devices and on switches' memory. A switch has a limited memory, when it is under attack, the number of flows received by the devices increase considerably, taking up all the storage capacity from the forwarding table. Many studies have been developed in order to create defense mechanisms to supply these vulnerabilities present in SDN architecture [7], [32]. Silva *et al.* [33] introduced a framework called ATLANTIC

(*Anomaly deTection and machine LeArNing Traffic classifIcation for software-defined networking*) for detecting, classifying and mitigating anomalous events in SDN networks. Garg and Garg [34] present an adaptive mechanism to update the policies dynamically to aggregate the flows inputs and to detect anomalies, so that the monitoring overcharge can be reduced and the anomalies can be detected more precisely. Mousavi and St-Hilaire [35] applied a technique to DDoS detection using entropy. The main goal of the authors is to detect an attack on its first stage, for the detection made at the beginning of the attack allows the application of mitigation policies before the controller is completely flooded with malicious packets.

Carvalho *et al.* [36] presents a new ecosystem to detect and mitigate DDoS attacks in SDN environments. The system proposed by the authors is composed by four stages: the first one is related to collection and storing of IP flow records; the second stage is the generation of a normal network profile based on the data collected using the ACODS method (*Ants Colony Optimization for Digital Signature*); the third stage corresponds to the detection of anomalies comparing the real network behavior to the generated profile using multinomial logistic regression (MLR) to detect suspicious events that differ from the expected behavior; finally, in the fourth stage mitigation policies are applied. The analysis of the traffic behavior for anomaly detection is done every 30 seconds. According to the results presented by the authors, the system proves to be efficient at the detection and mitigation of anomalous events stages.

Hamamoto *et al.* [37] proposed a system of anomalies detection applied to large scale networks. The authors used the DSNSF approach (*Digital Signature of Network Segment using Flow Analysis*) to generate behavior signatures of normal network behavior, applying GA (Genetic Algorithm). Furthermore, the Fuzzy logic was used along with DSNSFs generated to anomalous behaviors in those analyzed networks. It was used real data collected from the State University of Londrina by using sFlow to validate the proposed system. Three different anomalies were injected into the network's real data, using tools for simulation of anomalous events: DoS, DDoS, and Flash Crowd. The suggested system showed to be efficient, with a prediction rate above 96%. Different works also applied the DSNSF approach by using different techniques. However, the traffic characterizations on these works used an approach which analyses from two to four weeks of data for recognizing patterns and generation of normal profile in the network's regular environment. Moreover, a limitation presented by these works is that the attacks were detected in the period between 1 and 5 minutes. Unlike these works, the model proposed in this paper performs the prediction of normal traffic behavior by applying a sliding window and detecting anomalous events every second.

With the increase of the applications of image recognition, natural language processing, bioinformatics, the Deep Learning models had a fundamental role in solving these kinds of problems. Due to its huge capacity to extract knowledge in large scale from complex data, obtaining advantages on its results if we compared them to the traditional Machine Learning techniques [18], [21]. In Cybersecurity, the Deep Learning models are being applied in many different areas, for example intrusion detection [38], malware detection, spam detection [39], DDoS attacks detection [40], etc.

Li *et al.* [41] proposed a supervised Learning Machine mechanism of defense and detection of DDoS attacks in SDN network environments based on deep learning. The model presented by the authors consists of the following layers: input layer, forward recursive layer, reverse recursive layer, fully connected hidden layer, and output layer. At the construction of the model, it was employed Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN). According to the result gained by this model after detection, the SDN controller generates discard policies and sends them to the switches. For the test conduction, the ISCX dataset was employed to train the detection and verification model of defense architecture through DDoS attacks in real-time. According to the presented results, the defense method presented obtained an accuracy rate of 98%. However, the supervised learning to detect network attacks is a drawback, because the way the attackers executed the attacks is constantly being updated.

Tuan A Tang *et al.* [42] employed Deep Learning to detect anomalous flows in the SDN network. The authors suggested a Deep Neural Network (DNN) for a system that detects intruders, and the model was trained by using the NSL-KDD dataset. The dataset is made of 41 attributes. However, only a subset of 6 attributes were used. Through experiments, the suggested model only obtained an accuracy of 75.75%. The low amount of attributes influenced the low accuracy. Dey and Rahman [43] present a method of anomalies detection based on flows on the OpenFlow controller using DNN. The suggested model combined two approaches Gated Recurrent Unit and Long Short Term Memory (GRU-LSTM) to construct the intrusion detection system. Two methods of feature selection were employed for each anomaly analyzed to improve the model's performance, the NOVA F-Test and Recursive Feature Elimination. For the experiment process, the NSL-KDD dataset was also used. The experimental results demonstrated an accuracy of 87%. Shone *et al.* [38] suggested a new DL ensemble model for NIDS, which combines deep and shallow learning. The model combines Non-symmetric Deep Auto-Encoder and Random Forest. The data used for the test came from KDD CUP 99 and NSL-KDD datasets. The results showed an accuracy of 97.85%.

Despite this, many works available in the literature [17], [21], [25], [26], [28] for detecting DDoS attacks only evaluate a few types of DDoS attacks. Unlike these works, one of the main contributions presented by the system proposed in this paper is the detection of 12 types of DDoS attacks (e.g., NTP, DNS, LDAP, MSSQL, NetBIOS, SNMP, SSDP, UDP, UDP-Lag, WebD-DoS (ARME), SYN, and TFTP). In addition, the system proposed is capable of learning the normal

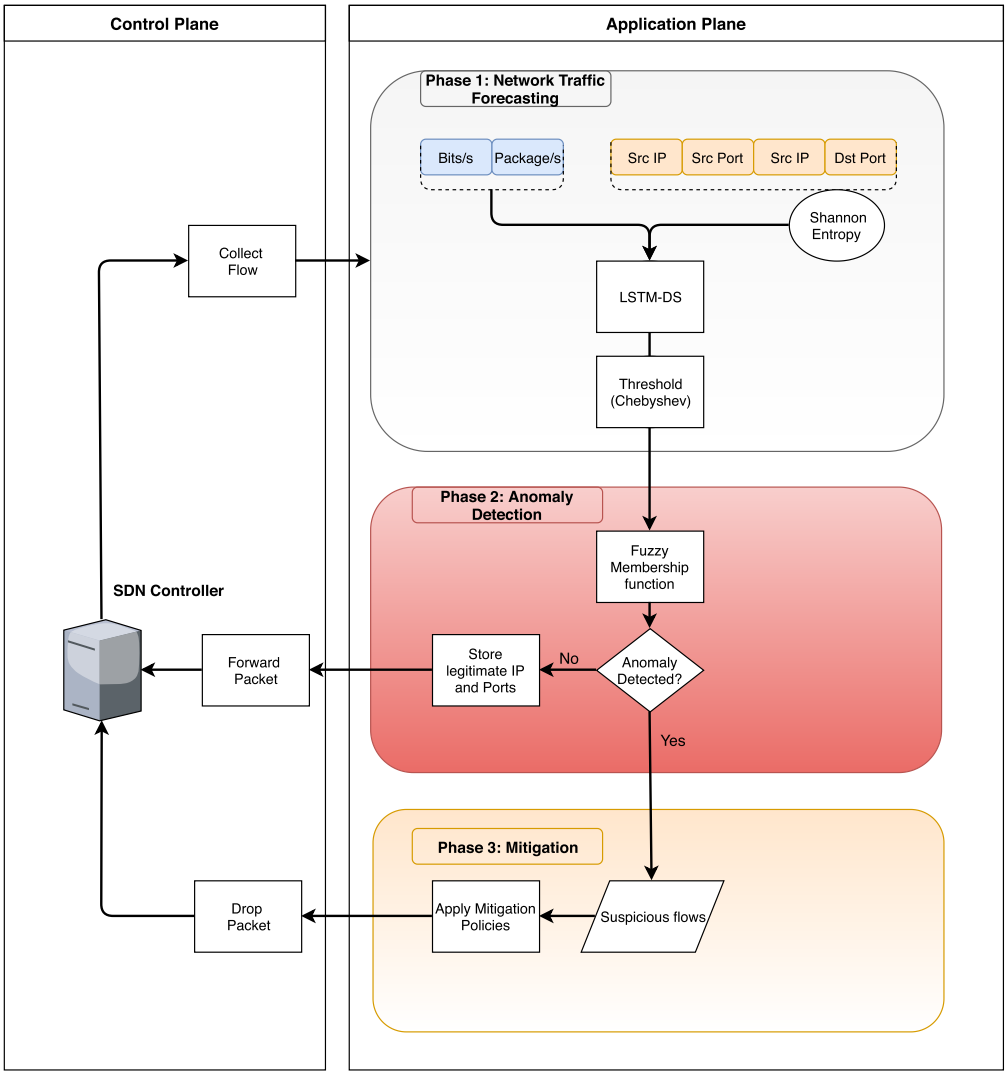


FIGURE 1. The proposed system architecture using LSTM and Fuzzy logic for Anomaly Detection and Mitigation.

behavior of the network, which is an advantage for detecting zero-day attacks.

III. THE SYSTEM PROPOSED

The system proposed in this paper has as its main goal the network traffic characterization, detection, and mitigation of DDoS attacks and Portscan in Software-defined networking environment. The system used as a principal the concept of Digital Signature of Network Segments (DSNS) introduced by Proença [20]. This concept applies an efficient technique to create a model that characterizes the network profile using historical data. The characterization proposed by Proença *et al*. was idealized for the traditional network environment and used a historical base from past traffic weeks containing MIB (Management Information Base) objects from management protocol SNMP (Simple Network Management Protocol). On the other hand, the characterization suggested in this paper uses IP flows attributes collected from the SDN controller, and the prediction of the network signature is made by employing a sliding window of the traffic. Consequently, the suggested system discards the use of a database to generate a signature. It is possible to recognize behavior from the normal profile of network that differs from the expected and helps in the anomaly detection stage methodology presented in this work.

The system of detection and mitigation of anomalies suggested in this work is divided in three phases:

1) Prediction of the normal behavior of the network's traffic and the definition of normality thresholds;
2) Application of the Fuzzy logic to determine if there are anomalies, using as a parameter the predicted traffic and the defined thresholds on the last stage;
3) Application of mitigation policies with the intention of taking countermeasures against the detected attacks, guarantying the network operation.

Fig. 1 illustrates the schematic diagram for operation of the anomaly detection and mitigation system suggested in this paper. The system was developed on the application

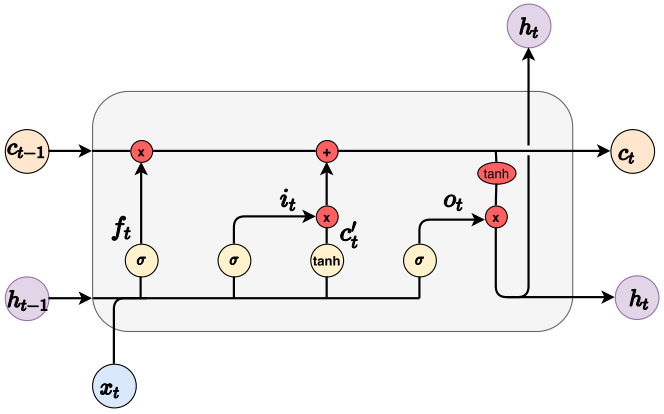


FIGURE 2. LSTM cell structure.

plane. The first stage is the traffic characterization, in which the flows attributes are pre-processed to predict the network traffic and the normal behavior signatures along with the normality thresholds. The next stage is the module of anomaly detection, in which the Fuzzy Inference system takes decision dynamically to determine the occurrence of anomalous events. When there is an anomaly, the IPs and ports that are in the analysis interval are considered suspicious. The third stage is responsible for the application of mitigation policies and receives as input suspicious flows determined on the last stage. In this set of flows, the mitigation module applies the most appropriate countermeasure to minimize the effects of an attack.

A. LONG SHORT-TERM MEMORY FOR NETWORK TRAFFIC FORECASTING

1) LONG SHORT-TERM MEMORY

In this subsection, some concepts about LSTM will be briefly introduced to assist in the understanding of the characterization module proposed in this work. Introduced by Hochreiter and Schmidhuber [44], LSTM a special architecture of recurrent artificial neural networks, with the capacity to learn long-term dependencies.

The structure of an LSTM cell is illustrated in Fig. 2. As observed, at each t time instant, the cell is controlled by various gates that can either maintain or reset the value according to the state of the gate. Three gates are applied on the cell, the forget gate (f_t), the input gate (i_t), and the output gate (o_t). Moreover, there is an entrance modulation gate called candidate value. The gates can be described as the following:

$$i_t = \sigma(W_{x,i}x_t + W_{i,h}h_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{f,i}x_t + W_{f,h}h_{t-1} + b_f) \quad (2)$$

$$o_t = \sigma(W_{o,i}x_t + W_{o,h}h_{t-1} + b_o) \quad (3)$$

$$c'_t = tanh(W_{c',i}x_t + W_{c',h}h_{t-1} + b_{c'}) \quad (4)$$

where W means the matrix of synaptic weight, b means the bias vectors, x_t is the actual input, c'_t is a vector with new candidates to be added to the actual state of the cell, h_{t-1} is the LSTM previous output in the time of instant $t-1$, $\sigma(\cdot)$ and $\tanh(\cdot)$ are the respective activation functions, Sigmoid and Tangent Hyperbolic. The first step on LSTM is to decide how much of the previous memory value will be removed from the state of the cell. This decision is made by the forget gate. The next stage is to determine how much of the new information will be stored, which is made by the input gate. Next, the state of the cell is used and defined with the following expression:

$$c_t = f_t \odot c_{t-1} + i_t \odot c'_t \quad (5)$$

in which $\odot$ denotes elementwise product. The LSTM output h_t is defined by:

$$h_t = o_t \odot tanh(c_t) \quad (6)$$

2) NETWORK TRAFFIC FORECASTING PHASE

The traffic prediction aims to generate the network's normal behavior signature, which is essential for the management and for the network security. The network characterization makes the decision of management related to possible problems that may occur more reliable and safer. To obtain a prediction close to the real behavior is an important step towards the detection of anomalous traffic, for the generated signature delimits the normality limits of a traffic sample at a certain moment on the network segment that is observed.

The characterizations of the signatures are generated from IP Flow data that are collected from the SDN network switches by the controller using an OpenFlow protocol. Among the attributes collected by the controller, the following attributes were selected: bits/s, packets/s, source IP address, destination IP address, source and destination ports. These flows attributes were analyzed and employed to previous works in the network traffic characterization of high speed and presented good results to describe and better understand the network behavior [45], [46]. Bytes and packets dimensions are quantitative attributes, which means volume attributes that are capable of supplying information related to the amount of information that is transported on the network. The others are nominal attributes and supply qualitative information that means these attributes allow to understand which devices are communicating with one another and which services are being accessed by them. The use of these attributes is fundamental to identifying possible attacks and is indispensable for the use of module mitigation to minimize the damage caused by an attack.

The IP and port attributes are nominal data and to carry out a quantitative analysis it is necessary to transform these attributes through entropy calculus. So, the Shannon Entropy was used in this work [47], it allows information from the concentration to be extracted and the prediction of these flow attributes. Ultimately, with the set of flow attributes $x = \{x_1, x_2, \ldots, x_n\}$ where x_i represents the sample's occurrence i at the interval of time. The entropy H to X is defined in

the Equation (7)

$$H(x) = -\sum_{i=1}^{N} \left(\frac{x_i}{S}\right) log_2 \left(\frac{x_i}{S}\right), \quad (7)$$

in which $S = \sum_{i=1}^{N} x_i$ is the sum of all occurrences present in the histogram.

It is possible to identify attacks by using entropy to characterize traffic. For example, during a DDoS attack occurrence, there is a concentration of the victim's IP address and destination port entropy; dispersion of the entropy of the source port due to multiple attackers using random source ports.

After guarantying that all flow attributes collected are presented in a quantitative way, a process of traffic signature generation starts. The problem with the network traffic prediction using LSTM could be defined as the following model. Consider at the instant of time t, the set of data $\mathbf{X} = (x_1, x_2, \ldots, x_d)$, where each x_i is a flow attribute vector defined as:

- x_1: bits/s
- x_2: packets/s
- x_3: source IP entropy
- x_4: destination IP entropy
- x_5: source Port entropy
- x_6: destination Port Entropy

Long Short-Term Memory neural networks are designed to handle with sequence due to their ability to maintain long term memory. In recent years, LSTM is widely used in time series prediction and has proven to be superior to traditional mathematical algorithms [48]–[50]. Besides, LSTM is a powerful technique that can represent the relationship between current and previous events and enhance network traffic forecasting.

In the approach to this work, LSTM was applied to mold the problem of univariate time series prediction. In this way, LSTM predicts the signature of normal network behavior. An LSTM was applied to each flow attribute defined previously, which means each LSTM will be responsible for predicting the signature of normal behavior for each attribute x_i. The LSTM model will learn a function that maps a sequence of n observations of previous inputs to an output observation [51]. For example, at the t instant, given an input sequence of n past observations that consists of the bits vector $x_1 = \{x_{t-n}, \ldots, x_{t-3}, x_{t-2}, x_{t-1}\}$, produces an output $y_1 = \{y_t\}$ which represents the behavior prediction to the flow bits attribute. Fig. 3 illustrates the LSTM model For Digital Signature (LSTM-DS) proposed in this paper.

Despite using 6 LSTM networks, one for each flow attribute, the training process of the network is an offline task. The computing cost for the training is high. However, it is not critical to its application [52]. Being so, during the operation stage with the trained LSTM networks, the prediction process of traffic is immediate. The Algorithm 1 illustrates the process of the LSTM-DS module operation.

The predicted traffic would not be the same as the real traffic. However, it is necessary to determine

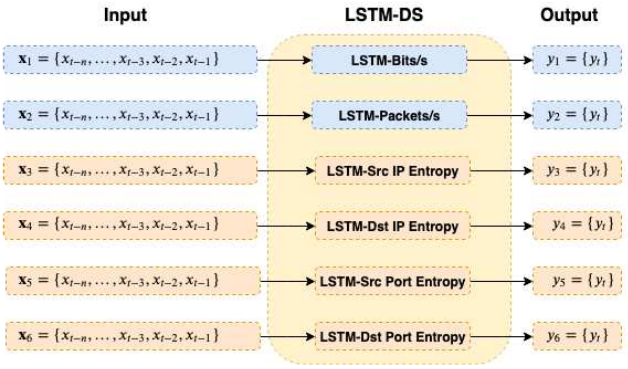


FIGURE 3. The proposed model for traffic forecasting using 6 LSTM.

Algorithm 1 LSTM-DS Operation Phase

Require: $\mathbf{X} = (x_1, x_2, \ldots, x_d)$
Ensure: $\mathbf{y} = (y_1, y_2, \ldots, y_d)$
1: $y_1 = \text{LSTM-bits}(x_1)$
2: $y_2 = \text{LSTM-Packets}(x_2)$
3: $y_3 = \text{LSTM-SrcIPEntropy}(x_3)$
4: $y_4 = \text{LSTM-DstIPEntropy}(x_4)$
5: $y_5 = \text{LSTM-SrcPortEntropy}(x_5)$
6: $y_6 = \text{LSTM-DstPortEntropy}(x_6)$

7: $\mathbf{y} = (y_1, y_2, y_3, y_4, y_5, y_6)$
8: **return y**
=0

thresholds between the predicted traffic and the real traffic. Bienaymé-Chebyshev's inequality is used to define this threshold between the predicted and the real one. Bienaymé-Chebyshev's inequality determines a limit of data percentage that lies in number k of standard deviations interval around the mean. The inequality can be applied to detect outliers [53] when the data distribution is unknown.

The equation for the Bienaymé-Chebyshev's inequality is represented in the Equation (8):

$$P(|\ X - \mu\ | \geq k\sigma) \leq \frac{1}{k^2}, \quad (8)$$

where X is a random variable, μ is the mean, $k > 0$ is the parameter of deviation and σ is the standard deviation. Defining the parameter $k = 4.47$ in Equation (8), the resulting probability will be equal to 0.05, which is the usual cut-off point for statistical significance to verify the discrepancy of a hypothesis in relation to the observed data [54].

B. FUZZY LOGIC FOR ANOMALY DETECTION

1) FUZZY LOGIC THEORY

In Classical logic, a proposition can only take values as true or false. On the other hand, the Fuzzy sets theory introduced a new concept, which means prepositions can take values from 0 to 1. This concept is called degrees of membership. Introduced by Zadeh in 1965 [55], the Fuzzy sets theory provides a mathematical tool capable of helping with decision taking in

an environment with imprecision variables, uncertainty and incomplete information.

A Fuzzy set can be defined as (S,f) where S is any set and f represents membership function. Every x element belongs to S, the $f(x)$ value defines the membership degree of x in the set(S,f). The x element is considered not included if $f(x) = 0$, totally included if $f(x) = 0$ and fuzzy member if $f(x) = 1$. An example of membership function is the Gaussiana, which is defined as:

$$f(x) = e^{\frac{-(x--m)^2}{2\sigma^2}} \tag{9}$$

where m is the mean and σ is the standard deviation of the S set.

According to Wu and Banzhaf [56], the Fuzzy logic is used to detect the anomalies in networks for two main reasons. The first one, the anomaly detection problems involve countless numeric attributes that are collected and derived statistically, which could cause a detection error. The second, the models that generate a normal profile of network behavior need to determine thresholds between the normal and anomalous behaviors. However, this interval is not well-defined and small changes (e.g., adversarial examples) on traffic behavior can cause false alarms. Considering these factors, the Fuzzy logic was used in this work to help with decision taking for anomaly detection.

2) ANOMALY DETECTION PHASE

The proposed model for anomaly detection in this work uses past traffic, the one predicted by LSTM and the Fuzzy logic. The first step is the ''fuzzification'' of sources for each one of the flow attributes being analyzed, applying the membership function. The membership function applied in this work is a modification of the Gaussian membership function, defined as:

$$f(y_t)_j = e^{\frac{-(x_t--y_t)^2}{2\hat{\sigma}_t^2}} \tag{10}$$

where x_t is the real traffic, y_t is the predicted traffic by LSTM and $\hat{\sigma}_t$ is the threshold generated by the Bienaymé-Chebyshev's inequality from the flow attribute j.

The Eq. 10 determines the membership degree of the normal traffic set. Therefore, to detect an anomaly we will apply its complement, defined as:

$$f_j' = 1 - f_j \tag{11}$$

The f_i' result represents the anomaly score of the flow attribute j. The anomaly scores are used to classify the traffic behavior to an instant data analysis. The process of ''defuzzification'' determines rather the traffic is ''normal'', ''Portscan'' or ''DDoS'', which are described in the following rules:

$$Rule \quad 1: \quad IF \quad \sum_{j=1}^{6} f_j' < \gamma \quad THEN \quad ''normal'' \tag{12}$$

$$Rule \quad 2: \quad IF \quad \sum_{j=1}^{6} f_j' \geq \gamma \quad AND \sum_{j=1}^{6} f_j' < \zeta \quad THEN \quad ''Portscan'' \tag{13}$$

$$Rule \quad 3: \quad IF \quad \sum_{j=1}^{6} f_j' \geq \zeta \quad THEN \quad ''DDoS'' \tag{14}$$

The values for γ and ζ scores were defined as 1.2362 and 3.3821, respectively. These values were rated by accuracy and are detailed on Section IV-C. Fig. 4 illustrates the anomaly score of all flow attributes during a day of network traffic analysis, which contains a DDoS and Portscan attack period. On the other hand, Fig. 5 illustrates the anomaly score sum of all six flow attributes. With the anomaly score calculated, the system can detect an attack based on the rules defined in (12), (13), and (14).

C. MITIGATION PHASE

The detection and identification of anomalies are essential stages that guaranty the operation and the services available throughout the network systems. After detecting an anomalous event, a mechanism must be used to minimize the effects caused by that event. The usual process to determine the effects caused by attacks is by mitigating, applying autonomic policies without the need for human interference, and aiming to ensure the network's resilience and operation. Thus, the proposed system consists of a module responsible for identifying the anomalous flows and mitigation policies are taken.

Mitigation policies are structured by using the **Event-Condition-Action** (ECA) model, which is considered adequate for the dynamic managing of policies. In this model, the **Event** refers to a specific anomaly and is associated with a set of rules. These rules are described as a set of **Conditions** that correspond to the context in which the anomaly took place. Finally, the **Action** is a countermeasure taken in relation to flows identified as malicious [57].

The main method used in applications for attack mitigation on SDN environments is to modify the flow table entry of the switch or to add a new flow entry. After detecting an attack, some characteristics must be identified, for example, source IP, IP destination, source port number, destination port number, and the kind of protocol. These characteristics help to identify the attacker and are fundamental for taking the countermeasures to minimize the damage an attack causes. A new entry on the flow table can be installed based on one or more of these characteristics, signaling that the packages that belong to the flow are from an attacker. Also, the actions taken can be the discard of these packets, anomalous traffic blockage and/or a honeypot redirection [58]

Based on the presented concepts, the mitigation module of the system is made by two policies to mitigate the detected anomalies. After the detection module's alarm goes off, the mitigation module takes action. The first step is to identify the suspect flows of the analysis interval. The identification of the suspected floes is made based on IP addresses analysis

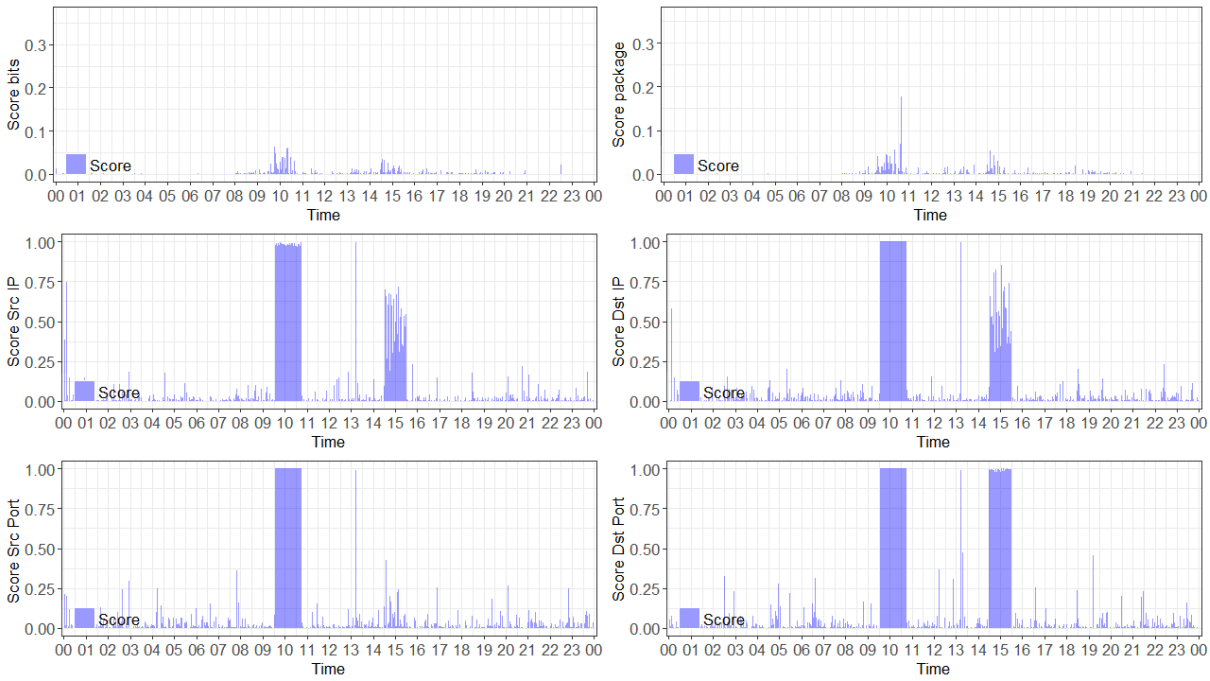


FIGURE 4. Anomaly score per flow attribute.

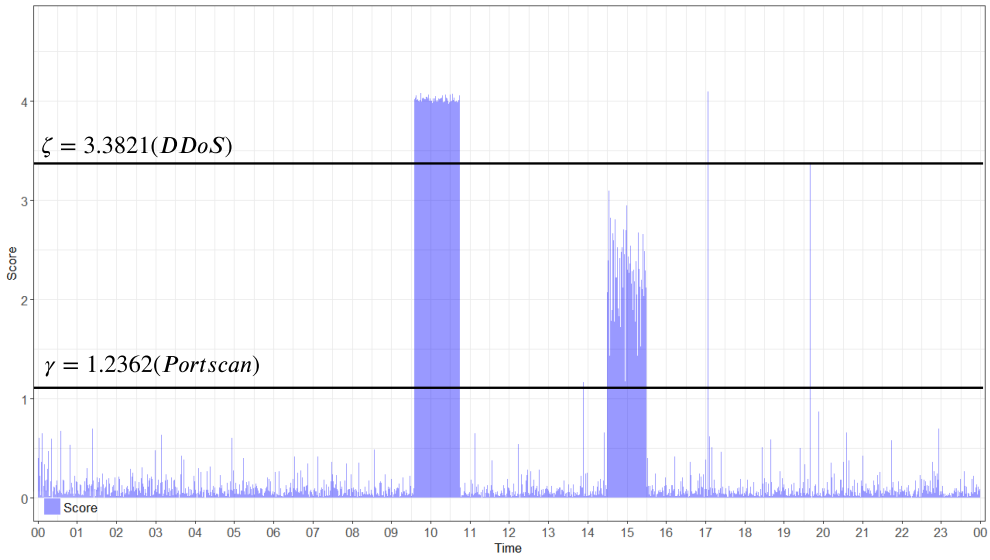


FIGURE 5. Anomaly score sum by six flows attributes.

and ports that make the anomalous interval. The ones that move toward the IP address that most receives flows are considered suspects.

By identifying the suspect flows, in case of an **Event** being launched by the detection module is a DDoS attack, a discard of the flows will be made based on the source IP addresses which appear more often on the suspected flows and which simultaneously have the same destination port. When launched **Event** is a Portscan attack, the process of identification of the attack is made by the origin IP address that presents the most variety of destination ports. This IP is considered an attacking one, and all its flows will be dropped. The process of mitigation is shown in the Algorithm 2.

There are anomalies that are not caused by malicious agents, but possess the same behavior of an attack. For example, a Flash crowd anomaly has the same characteristics of a DDoS attack, however, they are user performing legit requisitions. In the works of Giotis *et al.* [58] and Assis *et al.* [46], the authors suggest the implementation of a mechanism that maintains a list of IP flow attributes of legit users for a determined time of 5 minutes. This way, we also implemented a mechanism called *Safe List* that keeps a list of IP addresses

Algorithm 2 Mitigation Process

Require: Suspect flows
Ensure: Anomalous Packets Discard

```
if DDoS attack then
    Identify the destination IP address which receives the most flows
    Identify in those flows the attacker's IP address which have the same destination port
    if IPs and ports are on the Safe List then
        Forward packets
    else
        Drop packets
if Portscan attack then
    Identify the IP address that has received the most flows
    Identify in those flows the origin IP which presents the most variety of destination ports
    if IPs e ports are on the Safe List then
        Forward packets
    else
        Drop packets
```

and ports. So, this list is verified before starting the mitigation process.

IV. RESULTS AND DISCUSSION

The system was implemented by using the Python language and with development libraries for application of Deep Learning Keras and TensorFlow. The experiments were made in an environment with the following figures: Intel Core i5 2.21 GHz, 8 GB RAM and the operational system Windows 10. Default parameters as set, dropout = 0.2, loss function is MSE (Mean-Square Error), learning rate = 0.001, and optimizer was set as Adam proposed in [59], which is an adaptive learning rate optimization algorithm for training deep neural networks.

To demonstrate the effectiveness and efficiency of the system proposed, we applied tests applying from distinct scenarios. The test environment used in scenario 1 was a network topology with 120 hosts and the attacks were carried out in the periods of the day. In scenario 2 we used IP flows emulated from a public data called CICDDoS 2019 [60] from the Canadian Institute for Cybersecurity. This database contains different kinds of DDoS attacks and realistic traffic profiles.

A. SCENARIOS

The system performs a traffic behavior analysis each second. Therefore, the network flows must be collected in this time-lapse. Considering this analysis, in scenario 1, it was necessary to emulate the network behavior using the SDN Mininet network emulator [61], which allows the creation of realistic virtual networks consisting of controllers, hosts, links e switches on one single virtual machine. The Mininet uses light virtualization in the creation of personalized open code topologies and, it is broadly applied in this field to carry out researches and development of solutions for SDN environments. The experiments used a tool called Scapy [62] to inject traffic in the emulated network to make sure the emulated scenario is the closest it can get to a real SDN environment, with high rates of traffic going through the network.

TABLE 1. Information about the parameters of attacks in scenario 1.

Type of Attack	Attack Parameters
DDoS	Attackers: 16 Attacking IPs: 10.0.0.80 - 10.0.0.85 Victim IP address: 10.0.0.50 Time: 9:35 - 10:45
Portscan	Attacking IP: 10.0.0.10 Victim IP address: 10.0.0.24 Ports: 1 - 20000 Seconds to wait between packets: 0.15 Time: 14:30 - 15:30

Furthermore, to implement the anomaly detection and mitigation mechanism, we used the SDN controller Floodlight. A controller based on Java developed by BigSwich offers support to a wide variety of OpenFlow switches, virtual or physical, and it can cope with mixed networks, OpenFlow and no OpenFlow. The flows attributes used were: bit/s, packet/s, source IP entropy, destination IP entropy, source Port entropy and destination Port entropy.

Fig. 6 illustrates a topology emulated on scenario 1. The first scenario is formed by a topology in which its elements are distributed in the format of stars. This topology is made of central switches, in which six switches are connected. Each sub-network contains 20 hosts, totalizing 120 hosts. Two 24 hours day were emulated, that contains 86400 samples each day. The first day of emulation only contains samples of normal behavior of the network. This day was used in the LSTM training phase, as a semi-supervised training approach was used in its training. The second day of emulation was used to evaluate the system's operating performance in the detection and mitigation of attacks. Along with the emulation, two attacks were carried out with different intensities and duration time. There a DDoS attack and a Portscan attack. The information related to the parameters used in the attacks are shown in detail on TABLE 1. This dataset is available online.[1]

In scenario 2 we used the public dateset CICDDoS 2019 [60]. This set of data is distributed in two days, one for training and another for testing. The training set is made of 12 different kinds of DDoS attacks, being, NTP, DNS, LDAP, MSSQL, NetBIOS, SNMP, SSDP, UDP, UDP-Lag, WebDDoS (ARME), SYN e TFTP. The second day, the testing day, contains 6 kinds of DDoS attacks, being NetBios, LDAP, MSSQL, UDP, UDP-Lag and SYN. The flows attributes used were the same as scenario 1.

[1] http://www.uel.br/grupos/orion/datasets.html

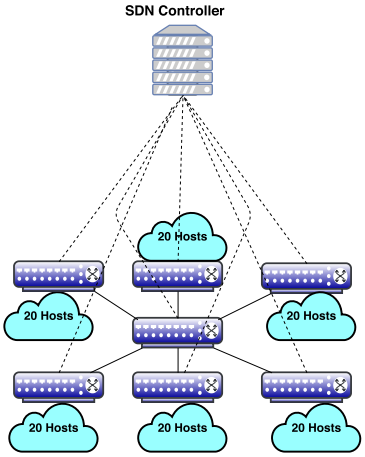


FIGURE 6. Network topology scenario 1 emulated on Mininet.

B. MÉTRICS AND TESTS

The tests applied aim to verify the efficiency of the suggested system, related to the modules that make it, detection and mitigation. The suggested module's performance results were analyzed using the following statistics metric [63]: precision, recall, false-positive rate.

1) precision: presents the percentage of intervals classified as anomalies, which are anomalies;
2) recall: measures how effective the model is in identifying the anomalous intervals about all the intervals;
3) false-positive rate: expresses a classification error, the traffic is identified as anomalous, but in fact, the traffic is normal.

These metrics can be easily calculated by the following equations:

$$\text{precision} = \frac{TP}{(TP + FP)} \tag{15}$$

$$\text{recall} = \frac{TP}{(TP + FN)} \tag{16}$$

$$\text{FPR} = \frac{FP}{(FP + FN)} \tag{17}$$

where, TP, TN, FP, FN mean true positive, true negative, false positive and false negative, respectively. Accuracy is a metric widely applied to anomaly detection works. However, it can lead to tendentious results where the dataset is unbalanced, which is the case of the data applied in this work. The dataset contains more normal samples than anomalous, and the system can classify all the samples correctly as normal and misclassify the anomalous samples, giving a tendentious result. The Precision metric can be used to solve this tendentious result and to emphasize the classification of correct anomalous samples.

The Receiver Operating Characteristics (ROC) [63] may be the combination of rates TP and FP, which gives a visual analysis of the system's capacity in detecting anomalous behaviors. However, to better quantify the efficiency between many classifiers, we analyze the area under the curve (AUC) of the ROC curve. The one with the highest value has the best ability to classify the samples. Therefore, AUC was applied to evaluate the proposed method with other models available in the literature.

TABLE 2. Contingency table (2 × 2).

	Positive test 2	Negative test 2	Sum row
Positive test 1	a	b	a+b
Negative test 1	c	d	c+d
Sum column	a+c	b+d	n

The efficiency of the module of mitigation was rated through the application of a statistic test called McNemar's Test, also through the dropped packet rate. The MacNemar Test a non-parametric test and its application is carried out through paired samples and nominal data. It is applied to contingency tables 2 × 2 with a dichotomous trace, which means, two behaviors (e.g., anomalous and normal) with the aim to verify if the marginal frequencies are equal or not [64] On TABLE 2 a generic example is illustrated of a contingency table 2 × 2 that presents the results of two tests in an sample of n individuals.

The null hypothesis indicates that the probabilities for each results are equal, that means, there was no change in the marginal frequencies and $p_a + p_b = p_a + p_c$ e $p_c + p_d = p_b + p_d$, where p_a, p_b, p_c, p_d indicate the theoretical probabilities of occurrences on the cells with the corresponding label. The null hypothesis and the alternative hypothesis are presented, respectively, as:

$$H_0 : p_b = p_c$$
$$H_1 : p_b \neq p_c$$

The MacNemar's test formula originated from the chi-square equation:

$$\chi^2 = \frac{(b-c)^2}{b+c} \tag{18}$$

χ^2 has a chi-squared distribution with a degree of freedom. If the result χ^2 is relevant, that means, that $p_b \neq p_c$ which means that the marginal frequencies are significantly different from one another the null hypothesis is rejected.

C. PARAMETERS EVALUATION

This section evaluates the results of the parameters used in the development of the proposed system. The first parameter to be looked into was the time step size used by the LSTM network in the traffic prediction phase. The values used for the test comprehend between 2 and 30 past samples of the traffic collected. The RMSE metric was used to determine the best time step size. The graphic present in Fig. 7 illustrates the values of RMSE obtained for each of the values evaluated. The time step size that presented the best result was equal to 5, with an RMSE value of 0.0445.

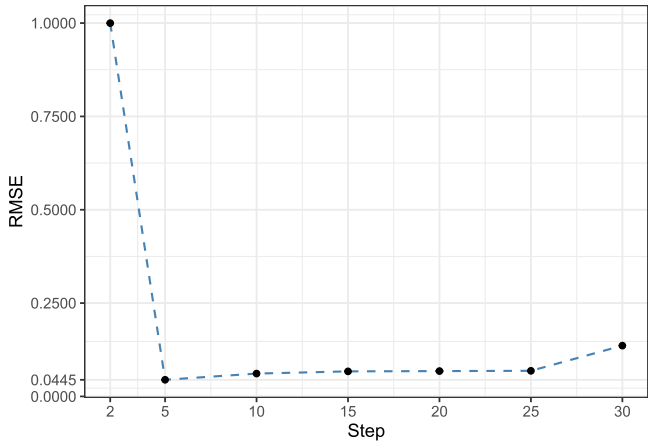


FIGURE 7. Time step size used for the LSTM traffic forecasting.

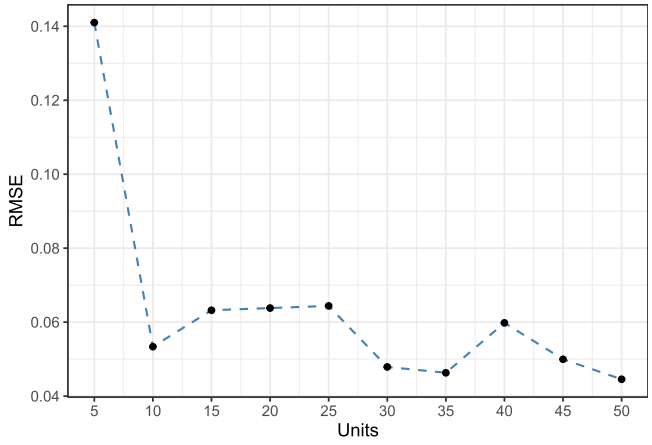


FIGURE 8. Number of units used for the LSTM traffic forecasting.

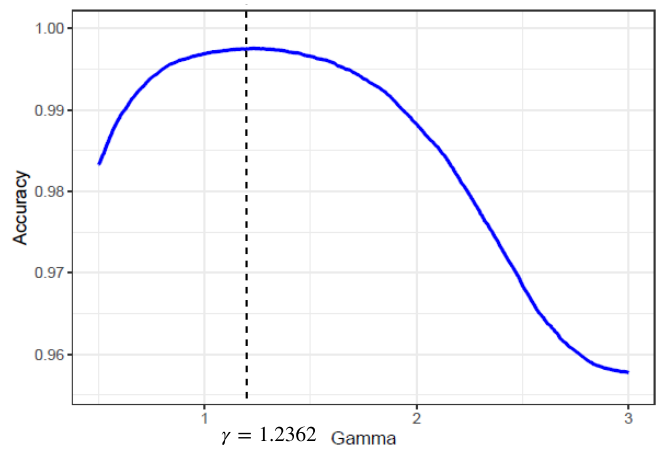


FIGURE 9. Accuracy evaluation for Gamma.

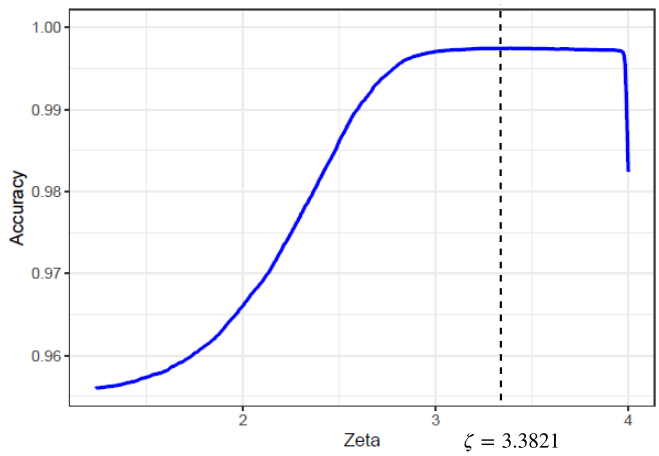


FIGURE 10. Accuracy evaluation for Zeta.

The next step was to define the number of hidden units. The evaluation of this parameter used the range of 5 to 100 units, and for each value the RMSE was evaluated. The number of units which presented the best result was 50 units, after which there was no significant improvement. In Fig. 8, we have a graphic with results obtained for each value of unit and its respective RMSE value.

The graphics present in Fig. 9 and Fig. 10 illustrate the evaluation of γ and ζ values to find the most adequate sum of anomaly score. The values were defined by varying γ and ζ and calculating its respective accuracies. The final value γ was defined with $argmax_{\gamma}(accuracy_{\gamma})$ and the ζ value was defined as $argmax_{\zeta}(accuracy_{\zeta})$. The score values for γ and ζ were defined as 1.2362 and 3.3821, respectively.

D. EVALUATION SCENARIO 1

To further validate our system, we compared our system with four other anomaly detection methods, which were applied to detect anomalies in SDN networks. The first method is the k-Nearest Neighbor (kNN) [65], a supervised classifier with a low computing cost, used to detect malicious events in a datacenter. The second method is the Multi-layer Perceptron (MLP) [66], an artificial neural network applied in the detection of DDoS attacks. Another method is based on Support Vector Machine (SVM) [67] to detect flooding attacks. We also compared it with LSTM-2 [68], which applied DL to detect DDoS attacks in the SDN environment. Finally, the recent method present in literature called Particle Swarm Optimization Digital Signature (PSO-DS) [46]. The heuristic method that used an unsupervised learning technique to detect DDoS and Portscan attacks on SDN networks.

TABLE 3. Information about the samples for each class.

Type of Traffic	Number of Samples
Normal	78420
DDoS	4380
Portscan	3600
Total	86400

To improve the comparison between the methods, on supervised approaches (kNN, SVM, MLP, and LSTM-2), we used a dataset for training that represents a day of network traffic data collection. This day is composed of normal traffic and by DDoS and Portscan attacks. The information for the number of samples to the classes are illustrated in Table 3.

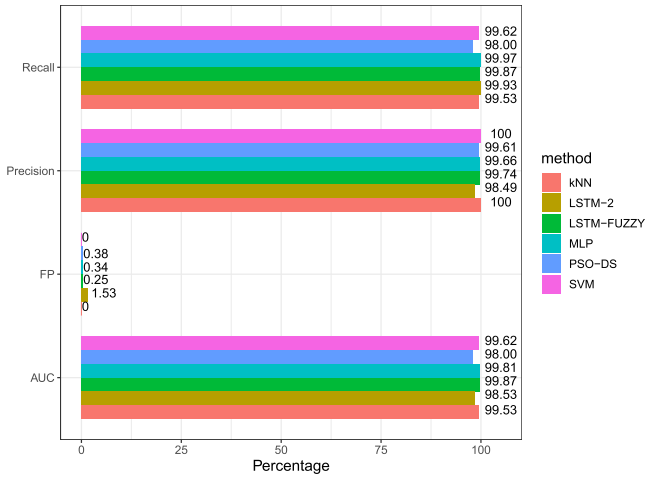


FIGURE 11. Detection results in the first scenario among LSTM-FUZZY and another methods.

A detailed analysis is illustrated in Fig. 11, where we present the metric results of compared methods. The LSTM-FUZZY presented a low false-positive rate, obtaining a value of 0.25%. The compared method LSTM-2 presented the highest false-positive rate, reaching 1.53%. On the other hand, the SVM and kNN methods didn't present false-positive rates. Regarding the recall and precision metrics, all the methods presented values superior to 98%. None of the methods reached better performance in all the metrics evaluated.

Fig. 12 presents the ROC curves, a visual comparison between the compared methods. Through the ROC curve, it is possible to determine which of the methods present the most adequate aptitude to detect anomalies. By analyzing the obtained results, it is clear that the LSTM-FUZZY approach obtained the best results among the other compared methods. The LSTM-FUZZY presented an AUC value of 99.87%, implying that the method presented the higher true positive rate with the lowest false-positive rate.

TABLE 4. Contingency table to evaluate the mitigation process on scenario 1.

	normal (after)	anomalous (after)
normal (before)	78323	86
anomalous (before)	7728	262

1) MITIGATION

From the alarms generated by the classification process of LSTM-FUZZY, mitigation policies were applied. Fig. 13 presents the traffic attributes in green without the application of mitigation and in blue is the traffic after the mitigation process. In the period between 9:45:00 and 10:35:00, we have a DDoS attack report, in this period we can see the increase of the packet and bits rate when the mitigation module is disabled, but by activating the module the traffic tends to go back to its normality due to discards of anomalous packets. The period of the Portscan attack between 14:30 and 15:30 causes minor changes in the traffic behavior, with the application of mitigation the affected attributes also go back to its normality.

In this scenario, the mitigation analysis through the McNemar's test and dropped packet rates were also applied. The significance level for the McNemar's test was $\alpha = 5\%$. The TABLE 4 offers information on the traffic classification between anomalous and normal before and after the mitigation process. By applying the test to the information on the table, the p-value results were lower than $2.2 \times e^{-16}$ that is smaller than the α value. Therefore, the null hypothesis in this scenario was also rejected. It indicates that there was indeed a difference in the frequencies. Thus, the mitigation was efficient in minimizing the threat effects. Also, the rate of anomalous packages dropped by the system was 99.88%. This result shows that almost all the anomalous packets were dropped.

E. EVALUATION SCENARIO 2

This scenario aims to evaluate the module of system detection, applying different kinds of DDoS attacks. As mentioned

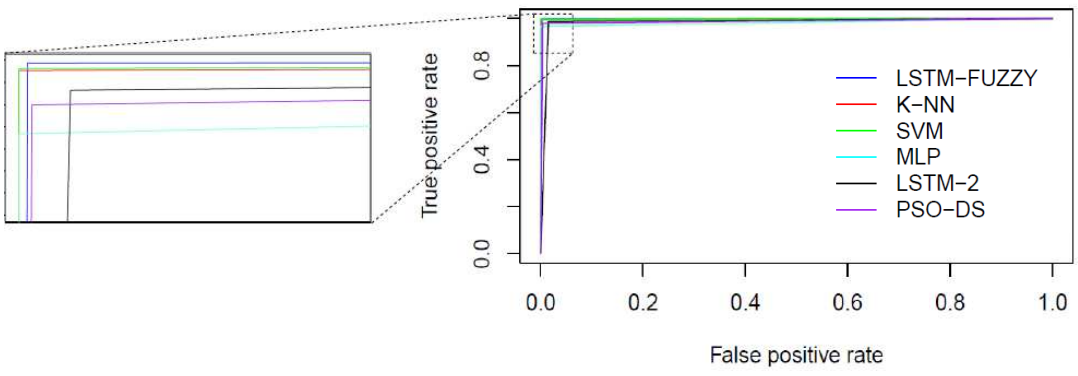


FIGURE 12. ROC curves of the methods compared scenario 1.

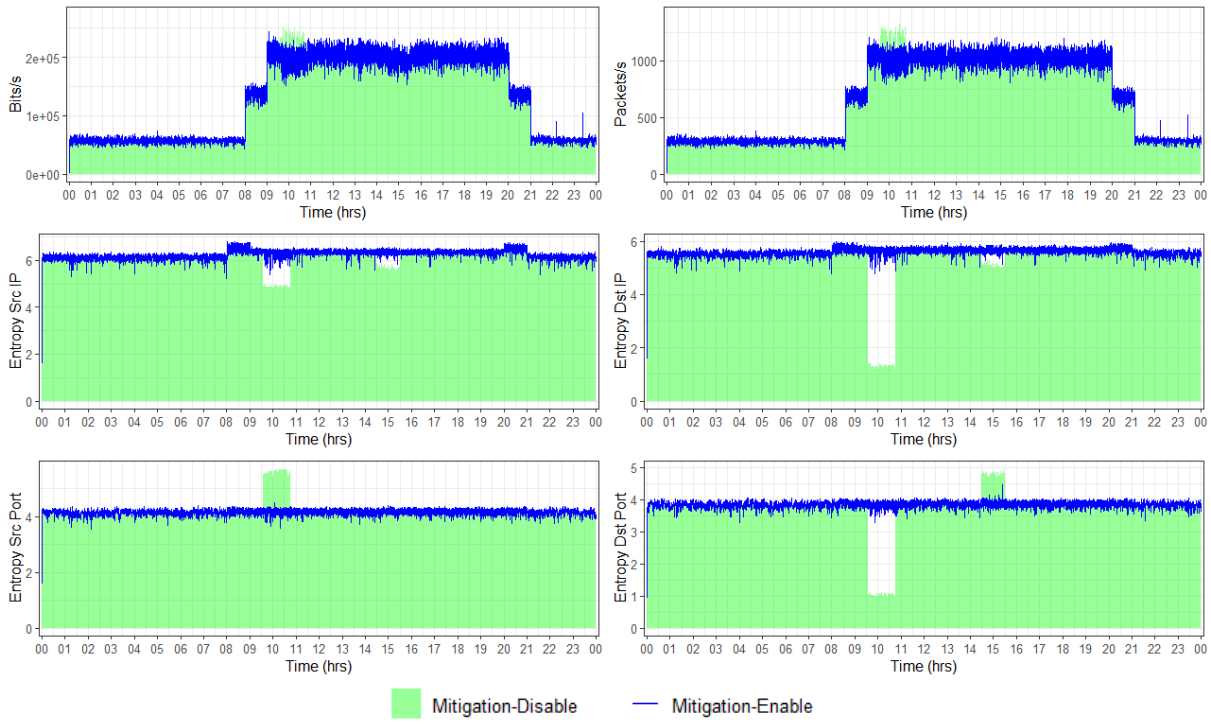


FIGURE 13. Graph showing non-mitigated traffic and mitigated traffic for intervals with anomaly on scenario 1.

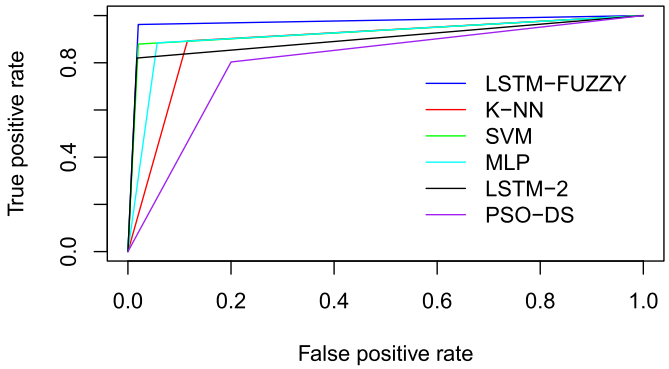


FIGURE 14. ROC curves of the methods compared scenario 2.

previously, the CICDDoS 2019 dataset [60], developed by the Canadian Institute for Cybersecurity, is made of two days (train and test). The training day is made of 12 kinds of DDoS attacks, and the test day contains six kinds of DDoS attacks.

As mentioned, the system suggested in this work does the analysis of traffic every 1 second. Thus, it was necessary to run a pre-process of the CICDDoS 2019 dataset to summarize the flows into groups of one-second intervals based on their timestamp feature. After grouping, we noted that all the intervals were made by only anomalous samples. To solve this problem, we separated the flows by anomalous and normal before the process of grouping in 1 second intervals.

However, the size of the flows samples containing DDoS attacks is superior to the normal data due to the characteristics of the attacks. It is not a problem to the LSTM-FUZZY, because on the training stage, the method only uses the normal samples to characterize the traffic, but it can generate overfitting for the methods SVM, kNN, MLP, and LSTM-2 that during the training a supervised approach is applied. To retain the characteristics and the representative of the applied data in the training, the solution applied to solve this problem was to sample the flows randomly for each kind of attack. Through empiric tests, for each kind of attack we selected a proportion of 5 times the normal flows. The set of training was reduced but it maintained enough sample quantity for the training process.

As executed in the first scenario, the efficiency of LSTM-FUZZY was compared to classic methods, kNN, SVM, MLP, LSTM-2, and PSO-DS. Fig. 15 illustrates the results of metrics obtained for each one of them. About the recall metric, it is clear that the LSTM-FUZZY obtained a performance superior to the other compared methods, obtaining a value of 93.13% for this metric, followed by LSTM-2, PSO-DS, kNN, MLP and SVM, which reached the rates of 90.53%, 89.66%, 89.27%, 87.92%, and 87.92%, respectively.

The next evaluated metric was the precision one, the LSTM-FUZZY again reached the best result, with a rate of 97.89%, the remaining ones were SVM, LSTM-2, MLP, kNN, and PSO-DS, reaching rates of 97.74%, 96.61%, 94.98%, 89.27%, and 81.19%, respectively. On the other hand, in comparison to the false-positive rate, the LSTM-FUZZY and the SVM reached the same rate of 2.2%, which

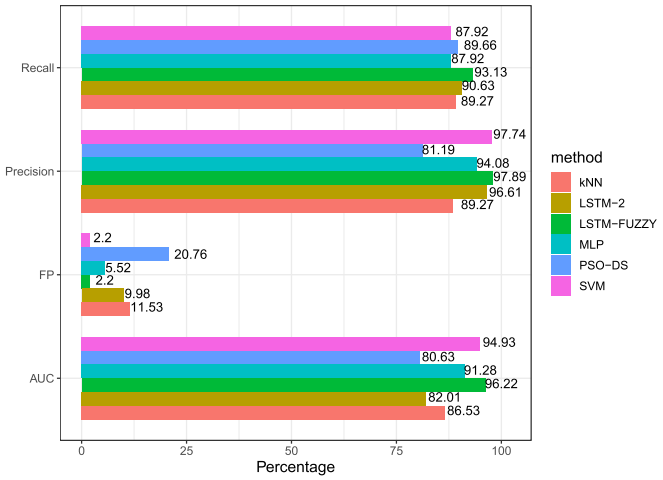


FIGURE 15. Detection results in the second scenario among LSTM-FUZZY and others methods.

could be considered a good result. After it was the MLP, LSTM-2, kNN and the PSO-DS, with retrospective values of 5.52%, 9.98%, 11.53%, and 20.76%. The proposed system showed superior results to the other compared methods, except for the SVM that obtained similar results. However, when using the ROC curve it was possible to observe the improvement between the compared methods more clearly. Despite the similar results, the performance applied by the proposed system is a significant improvement, as current computer networks operate with links with high transmission rates. Over a day of network operation, a small percentage of undetected attacks could cause damage to its operation. For instance, in October 2016, a DDoS attack with 100 thousand malicious endpoints surpassed a bandwidth of 1.2 Tbps [69]. As the outcomes presented in the first scenario, the LSTM-FUZZY method also fared better on the average than the other compared methods on the second scenario, achieving promising test outcomes that make it an efficient technique on detecting different kinds of DDoS attacks.

Just as in the previous scenario, the ROC curve was used to determine which method presented the best performance in detecting attacks. Fig. 14 presents the visual analysis of the ROC curve. Through AUC, we can see that the LSTM-FUZZY was the one that reached the best balance between the true-positive rates and the false-positive rates, reaching a value of 96.22%. Followed by the SVM, MLP, kNN, LSTM-2, and PSO-DS with the following values 94.93%, 91.28%, 86.53%, 82.01%, and 80.63.

1) MITIGATION

In this scenario, we evaluated the efficiency to mitigate the DDoS attacks from CICDDoS 2019 dataset. Fig. 16 presents the traffic behavior from the test day where there is the DDoS attack report with the mitigation module deactivated and compares its behavior when the mitigation is activated. The traffic generated without the application of mitigation policy is represented in the green area, and the blue line shows the traffic after the application of mitigation against the DDoS attacks. Visually it is possible to see when the attacks are

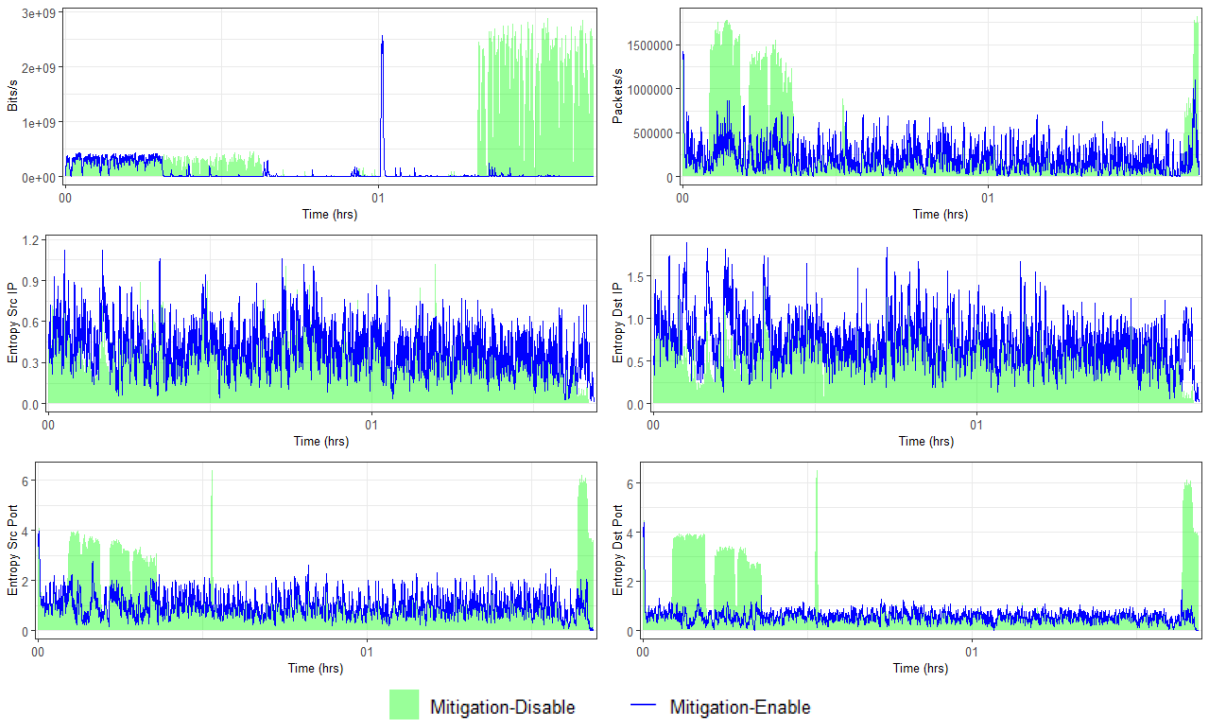


FIGURE 16. Traffic analysis with mitigation module disable and enable on test day from CICDDoS 2019 dataset.

TABLE 5. Contingency table applied for evaluation of mitigation on scenario 2.

	normal (after)	anomalous (after)
normal (before)	4811	136
anomalous (before)	2175	68

mitigated, the attribute values being analyzed return to its expected behavior.

The MacNemar's test was applied with a level of significance of $\alpha = 5\%$ and the null hypothesis that the marginal frequencies are equal. After applying the test on the contingency TABLE 5, the p-value result was $2.2 \times e^{-16}$ that is lower than the value α. Thus, the null hypothesis is rejected, which indicates that there was a difference in the marginal frequencies, and the mitigation was effective. Moreover, the anomalous packets rates discarded was 99.20%, which implies that the majority of the anomalous packages were mitigated.

V. CONCLUSION

In this work, we presented a modular system for detection and mitigation of anomalies in SDN networks. The system is made of three modules where its activities are carried out in an automatized way to make monitoring, detection, and mitigation of attacks easier. In the first module, responsible for the characterization of traffic, we developed a new approach to predict the normal behavior of the network operation, applying an approach of Long Short-Term Memory (LSTM) semi-supervised using IP flows. In the second module, we proposed a mechanism to recognize attacks, through the application of Bienaymé-Chebyshev's inequality along with the Fuzzy logic. Finally, in the third module, we applied automatized mitigation policies to minimize the damage caused by attacks and to maintain the requirement of network operation.

To validate the development system, we employed two scenarios with distinct characteristics. In the first scenario, we used emulated SDN data, using the Mininet emulator and the Floodlight controller, containing periods of DDoS and Portscan attacks. In the second scenario, we used a public dataset called CICDDoS 2019. This dataset is made of 12 kinds of different DDoS attacks. To test the detection module, we compared the LSTM-FUZZY with the other methods present in the literature, SVM, kNN, MLP, LSTM-2, and PSO-DS. In both scenarios we compared the performance between the suggested method and the others. According to the results presented, the LSTM-FUZZY presented a superior performance compared to the others, reaching a low false-positive rate and high precision, recall, and AUC rates.

In the first scenario, we applied mitigation policies based on the kind of attack identified by the detection module. In this module, we identified the suspect flows, based on the analysis of IP addresses and ports that make the anomalous interval. The flows identified as suspects were dropped. Through the McNemar's test and dropped anomalous packets rate, it was shown that the module obtained a satisfactory performance, minimizing the effects of the attacks.

The LSTM power to learn to extract short and long-term patterns allowed the application to predict the normal behavior of the network traffic. The module produced adequate predictions close to real traffic behavior, and it was possible to apply them in the detection stage. The Fuzzy Logic characteristics allowed anomaly detection in an unsupervised way, implying that the system does not need labeled data. The advantage of using this technique makes the system operation easier and discards the need to use a labeled dataset, which demands much work and could be full of human errors. Moreover, the Fuzzy Logic acts on the detection of different DDoS attacks with a low false-positive rate, allowing the system to act on the present SDN environment with high accuracy to detect and low false alarms.

The results obtained show that the modules that made the proposed system were efficient, meeting the goals assigned to each one of them. The execution of the activities carried out by the system is automatic, which means the process of monitoring, identification of adverse events, and the countermeasures are carried out without the need for human interference. The monitoring and managing of the network is a complex activity. The application of an autonomous system helps the assigned tasks to the administrator to maintain and guaranty the network's operation to its fullest. Hence, the system developed in this work can be applied to collaborate and facilitate management procedures and to guaranty the availability of the services offered.

The modular architecture of the system allows the maintenance and adaptation of other techniques to characterize traffic, detection, and mitigation of anomalies in SDN environments. This characteristic allows the adaptation of the system as the network dynamics change, and new security demands emerge. Thus, future works can explore other vulnerabilities and incorporate mitigation policies to meet new demands that might emerge in SDN network environments. Another point that could be extended is the exploration of more tests in other scenarios with different types of topology and attacks.

REFERENCES

[1] R. Khan, P. Kumar, D. N. K. Jayakody, and M. Liyanage, "A survey on security and privacy of 5G technologies: Potential solutions, recent advancements, and future directions," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 196–248, 1st Quart., 2020.

[2] A. Rego, L. Garcia, S. Sendra, and J. Lloret, "Software defined network-based control system for an efficient traffic management for emergency situations in smart cities," *Future Gener. Comput. Syst.*, vol. 88, pp. 243–253, Nov. 2018.

[3] A. A. Barakabitze, A. Ahmad, R. Mijumbi, and A. Hines, "5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges," *Comput. Netw.*, vol. 167, Feb. 2020, Art. no. 106984.

[4] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[5] T. Das, V. Sridharan, and M. Gurusamy, "A survey on controller placement in SDN," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 472–503, 1st Quart., 2020.

[6] O. Salman, I. Elhajj, A. Chehab, and A. Kayssi, "IoT survey: An SDN and fog computing perspective," *Comput. Netw.*, vol. 143, pp. 221–246, Oct. 2018.

[7] M. B. M. Noor and W. H. Hassan, "Current research on Internet of Things (IoT) security: A survey," *Comput. Netw.*, vol. 148, pp. 283–294, Jan. 2019.
[8] Z. Li, W. Xing, S. Khamaiseh, and D. Xu, "Detecting saturation attacks based on self-similarity of OpenFlow traffic," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 1, pp. 607–621, Mar. 2020.
[9] M. Jonker, A. King, J. Krupp, C. Rossow, A. Sperotto, and A. Dainotti, "Millions of targets under attack: A macroscopic characterization of the DoS ecosystem," in *Proc. Internet Meas. Conf.*, New York, NY, USA, Nov. 2017, pp. 100–113.
[10] T. A. Pascoal, I. E. Fonseca, and V. Nigam, "Slow denial-of-service attacks on software defined networks," *Comput. Netw.*, vol. 173, May 2020, Art. no. 107223.
[11] V. Varadharajan, K. Karmakar, U. Tupakula, and M. Hitchens, "A policy-based security architecture for software-defined networks," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 4, pp. 897–912, Apr. 2019.
[12] V. Varadharajan and U. Tupakula, "Counteracting attacks from malicious end hosts in software defined networks," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 1, pp. 160–174, Mar. 2020.
[13] C. Yoon, T. Park, S. Lee, H. Kang, S. Shin, and Z. Zhang, "Enabling security functions with SDN: A feasibility study," *Comput. Netw.*, vol. 85, pp. 19–35, Jul. 2015.
[14] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 602–622, 1st Quart., 2016.
[15] K. Kalkan, G. Gur, and F. Alagoz, "Defense mechanisms against DDoS attacks in SDN environment," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 175–179, Sep. 2017.
[16] C. Zhang, G. Hu, G. Chen, A. K. Sangaiah, P. Zhang, X. Yan, and W. Jiang, "Towards a SDN-based integrated architecture for mitigating IP spoofing attack," *IEEE Access*, vol. 6, pp. 22764–22777, 2018.
[17] N. Moustafa, J. Hu, and J. Slay, "A holistic review of network anomaly detection systems: A comprehensive survey," *J. Netw. Comput. Appl.*, vol. 128, pp. 33–55, Feb. 2019.
[18] S. Mahdavifar and A. A. Ghorbani, "Application of deep learning to cybersecurity: A survey," *Neurocomputing*, vol. 347, pp. 149–176, Jun. 2019.
[19] W. Wu, R. Li, G. Xie, J. An, Y. Bai, J. Zhou, and K. Li, "A survey of intrusion detection for in-vehicle networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 3, pp. 919–933, Mar. 2020.
[20] M. L. Proenca, B. B. Zarpelao, and L. S. Mendes, "Anomaly detection for network servers using digital signature of network segment," in *Proc. Adv. Ind. Conf. Telecommun./Service Assurance Partial Intermittent Resour. Conf./E-Learn. Telecommun. Workshop (AICT/SAPIR/ELETE)*, Jul. 2005, pp. 290–295.
[21] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli, "A detailed investigation and analysis of using machine learning techniques for intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 686–728, 1st Quart., 2019.
[22] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for IoT security based on learning techniques," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2671–2701, 3rd Quart., 2019.
[23] R. K. Malaiya, D. Kwon, S. C. Suh, H. Kim, I. Kim, and J. Kim, "An empirical evaluation of deep learning for network anomaly detection," *IEEE Access*, vol. 7, pp. 140806–140817, 2019.
[24] Y. Zeng, H. Gu, W. Wei, and Y. Guo, "*Deep − full − range*: A deep learning based network encrypted traffic classification and intrusion detection framework," *IEEE Access*, vol. 7, pp. 45182–45190, 2019.
[25] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou, and C. Wang, "Machine learning and deep learning methods for cybersecurity," *IEEE Access*, vol. 6, pp. 35365–35381, 2018.
[26] A. Aldweesh, A. Derhab, and A. Z. Emam, "Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues," *Knowl.-Based Syst.*, vol. 189, Feb. 2020, Art. no. 105124.
[27] F. A. Khan, A. Gumaei, A. Derhab, and A. Hussain, "TSDL: A two-stage deep learning model for efficient network intrusion detection," *IEEE Access*, vol. 7, pp. 30373–30385, 2019.
[28] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.
[29] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *J. Inf. Secur. Appl.*, vol. 50, Feb. 2020, Art. no. 102419.
[30] C. Gkountis, M. Taha, J. Lloret, and G. Kambourakis, "Lightweight algorithm for protecting SDN controller against DDoS attacks," in *Proc. 10th IFIP Wireless Mobile Netw. Conf. (WMNC)*, Sep. 2017, pp. 1–6.
[31] A. AlEroud and I. Alsmadi, "Identifying cyber-attacks on software defined networks: An inference-based intrusion detection approach," *J. Netw. Comput. Appl.*, vol. 80, pp. 152–164, Feb. 2017.
[32] G. Fernandes, J. J. P. C. Rodrigues, L. F. Carvalho, J. F. Al-Muhtadi, and M. L. Proença, "A comprehensive survey on network anomaly detection," *Telecommun. Syst.*, vol. 70, no. 3, pp. 447–489, Mar. 2019.
[33] A. S. Da Silva, J. A. Wickboldt, L. Z. Granville, and A. Schaeffer-Filho, "ATLANTIC: A framework for anomaly traffic detection, classification, and mitigation in SDN," in *Proc. NOMS-IEEE/IFIP Netw. Oper. Manage. Symp.*, Apr. 2016, pp. 27–35.
[34] G. Garg and R. Garg, "Detecting anomalies efficiently in SDN using adaptive mechanism," in *Proc. 5th Int. Conf. Adv. Comput. Commun. Technol.*, Feb. 2015, pp. 367–370.
[35] S. M. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against SDN controllers," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2015, pp. 77–81.
[36] L. F. Carvalho, T. Abrão, L. D. S. Mendes, and M. L. Proença, "An ecosystem for anomaly detection and mitigation in software-defined networking," *Expert Syst. Appl.*, vol. 104, pp. 121–133, Aug. 2018.
[37] A. H. Hamamoto, L. F. Carvalho, L. D. H. Sampaio, T. Abrao, and M. Proença, "Network anomaly detection system using genetic algorithm and fuzzy logic," *Expert Syst. Appl.*, vol. 92, pp. 390–402, Feb. 2018.
[38] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.
[39] P. K. Roy, J. P. Singh, and S. Banerjee, "Deep learning to filter SMS spam," *Future Gener. Comput. Syst.*, vol. 102, pp. 524–533, Jan. 2020.
[40] X. Yuan, C. Li, and X. Li, "DeepDefense: Identifying DDoS attack via deep learning," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, May 2017, pp. 1–8.
[41] C. Li, Y. Wu, X. Yuan, Z. Sun, W. Wang, X. Li, and L. Gong, "Detection and defense of DDoS attack-based on deep learning in OpenFlow-based SDN," *Int. J. Commun. Syst.*, vol. 31, no. 5, p. e3497, Mar. 2018.
[42] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *Proc. Int. Conf. Wireless Netw. Mobile Commun. (WINCOM)*, Oct. 2016, pp. 258–263.
[43] S. K. Dey and M. M. Rahman, "Flow based anomaly detection in software defined networking: A deep learning approach with feature selection method," in *Proc. 4th Int. Conf. Electr. Eng. Inf. Commun. Technol. (iCEE-iCT)*, Sep. 2018, pp. 630–635.
[44] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[45] E. H. M. Pena, S. Barbon, J. J. P. C. Rodrigues, and M. L. Proenca, "Anomaly detection using digital signature of network segment with adaptive ARIMA model and paraconsistent logic," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jun. 2014, pp. 1–6.
[46] M. V. O. De Assis, M. P. Novaes, C. B. Zerbini, L. F. Carvalho, T. Abrao, and M. L. Proenca, "Fast defense system against attacks in software defined networks," *IEEE Access*, vol. 6, pp. 69620–69639, 2018.
[47] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948.
[48] B. Yang, S. Sun, J. Li, X. Lin, and Y. Tian, "Traffic flow prediction using LSTM with feature enhancement," *Neurocomputing*, vol. 332, pp. 320–327, Mar. 2019.
[49] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *Proc. 41st Int. ACM SIGIR Conf. Res. Development Inf. Retr.*, New York, NY, USA, 2018, pp. 95–104.
[50] J. Bhatia, R. Dave, H. Bhayani, S. Tanwar, and A. Nayyar, "SDN-based real-time urban traffic analysis in VANET environment," *Comput. Commun.*, vol. 149, pp. 162–175, Jan. 2020.
[51] M. Munir, S. A. Siddiqui, A. Dengel, and S. Ahmed, "DeepAnT: A deep learning approach for unsupervised anomaly detection in time series," *IEEE Access*, vol. 7, pp. 1991–2005, 2019.
[52] X. Qing and Y. Niu, "Hourly day-ahead solar irradiance prediction using weather forecasts by LSTM," *Energy*, vol. 148, pp. 461–468, Apr. 2018.

[53] B. G. Amidan, T. A. Ferryman, and S. K. Cooley, "Data outlier detection using the Chebyshev theorem," in *Proc. IEEE Aerosp. Conf.*, Mar. 2005, pp. 3–8.
[54] C. Taylor and J. Alves-Foss, "An empirical analysis of NATE: Network analysis of anomalous traffic events," in *Proc. Workshop New Secur. Paradigms*, New York, NY, USA, 2002, pp. 18–26.
[55] U. R. Rosyara, D. Vromman, and E. Duveiller, "Canopy temperature depression as an indication of correlative measure of spot blotch resistance and heat stress tolerance in spring wheat," *J. Plant Pathol.*, vol. 90, no. 1, pp. 103–107, 2008.
[56] S. X. Wu and W. Banzhaf, "The use of computational intelligence in intrusion detection systems: A review," *Appl. Soft Comput.*, vol. 10, no. 1, pp. 1–35, Jan. 2010.
[57] R. Sahay, G. Blanc, Z. Zhang, K. Toumi, and H. Debar, "Adaptive policy-driven attack mitigation in SDN," in *Proc. 1st Int. Workshop Secur. Dependability Multi-Domain Infrastruct. (XDOMO)*, New York, NY, USA, 2017, pp. 4:1–4:6.
[58] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments," *Comput. Netw.*, vol. 62, pp. 122–136, Apr. 2014.
[59] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, Y. Bengio and Y. LeCun, Eds. San Diego, CA, USA, May 2015.
[60] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in *Proc. Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2019, pp. 1–8.
[61] (2019). Mininet Team. *Mininet Overview*. Accessed: Dec. 3, 2019. [Online]. Available: http://mininet.org/overview/
[62] P. Biondi. (2019). *Scapy*. Accessed: Dec. 3, 2019. [Online]. Available: http://www.secdev.org/projects/scapy/
[63] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, Jun. 2006.
[64] X. Sun and Z. Yang, "Generalized McNemar's test for homogeneity of the marginal distributions," in *Proc. SAS Global Forum*, vol. 382, 2008, pp. 1–10.
[65] P. Xiao, W. Qu, H. Qi, and Z. Li, "Detecting DDoS attacks against data center with correlation analysis," *Comput. Commun.*, vol. 67, pp. 66–74, Aug. 2015.
[66] M. Wang, Y. Lu, and J. Qin, "A dynamic MLP-based DDoS attack detection method using feature selection and feedback," *Comput. Secur.*, vol. 88, Jan. 2020, Art. no. 101645.
[67] T. V. Phan, T. Van Toan, D. Van Tuyen, T. T. Huong, and N. H. Thanh, "OpenFlowSIA: An optimized protection scheme for software-defined networks from flooding attacks," in *Proc. IEEE 6th Int. Conf. Commun. Electron. (ICCE)*, Jul. 2016, pp. 13–18.
[68] R. Priyadarshini and R. K. Barik, "A deep learning based intelligent framework to mitigate DDoS attack in fog environment," *J. King Saud Univ.-Comput. Inf. Sci.*, early access, Apr. 24, 2019, doi: 10.1016/j.jksuci.2019.04.010.
[69] E. Viegas, A. Santin, A. Bessani, and N. Neves, "BigFlow: Real-time and reliable anomaly-based intrusion detection for high-speed networks," *Future Gener. Comput. Syst.*, vol. 93, pp. 473–485, Apr. 2019.

MATHEUS P. NOVAES received the master's degree in computer science from the State University of Londrina (UEL), Brazil, where he is currently pursuing the Ph.D. degree with the Electrical Engineering Department. He has been a member of the Research Group Computer Networks and Data Communication, Computer Science Department, UEL. His research interest includes management and security of computer networks.

LUIZ F. CARVALHO received the master's degree in computer science from the State University of Londrina, in 2014, and the Ph.D. degree in electrical engineering and telecommunications from the State University of Campinas, in 2018. He has experience in computer science with emphasis in computer networks and is part of the Research Group Computer Networks and Data Communication. His main research interests include management and security of computer networks and software-defined networks.

JAIME LLORET (Senior Member, IEEE) received the M.Sc. degrees in physics and electronic engineering from the University of Valencia, Valencia, Spain, in 1997 and 2003, respectively, and the Ph.D. degree in telecommunication engineering (Dr.Ing.) from the Polytechnic University of Valencia, Valencia, in 2006.

He is currently an Associate Professor with the Department of Communications, Polytechnic University of Valencia.

Dr. Lloret was the Internet Technical Committee Chair, from 2014 to 2015. He is also the Chair of IEEE 1907.1. He is also the Director of the Research Institute Integrated Management Coastal Research Institute (IGIC) and the Head of the Innovation Group Active and collaborative techniques and use of technologic resources in the education (EITACURTE). He has been the General Chair of 36 international workshops and conferences. He is also the Co-Editor-in-Chief of *Ad Hoc and Sensor Wireless Networks* and the Editor-in-Chief of *Network Protocols and Algorithms*.

MARIO LEMES PROENÇA, Jr. received the M.Sc. degree in computer science from the Informatics Institute, Federal University of Rio Grande do Sul (UFRGS), in 1998, and the Ph.D. degree in electrical engineering and telecommunications from the State University of Campinas (UNICAMP), in 2005. He is currently an Associate Professor and the Leader of the Research Group that studies computer networks in the Computer Science Department, State University of Londrina (UEL), Brazil. He has authored or coauthored over 100 articles in refereed international journals and conferences, books chapters, and one software register patent. His research interests include computer networks, network operations, management and security, and IT governance. He has supervised 14 M.Sc. and three Ph.D. students. He has been a Master's Supervisor in computer science with the State University of Londrina and a Ph.D. Supervisor with the Department of Electrical Engineering, UEL.

Apêndice B: Adversarial Deep Learning approach detection and defense against DDoS attacks in SDN environments

Future Generation Computer Systems 125 (2021) 156–167

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Adversarial Deep Learning approach detection and defense against DDoS attacks in SDN environments

Matheus P. Novaes [a], Luiz F. Carvalho [b], Jaime Lloret [c,*], Mario Lemes Proença Jr. [d]

[a] Electric Engineering Department, State University of Londrina (UEL), Londrina, Paraná, Brazil
[b] Computer Engineering Department, Federal Technology University of Paraná (UTFPR), Apucarana, Paraná, Brazil
[c] Integrated Management Coastal Research Institute, Universitat Politecnica de Valencia, Valencia, Spain
[d] Computer Science Department, State University of Londrina (UEL), Londrina, Paraná, Brazil

ARTICLE INFO

Article history:
Received 23 January 2021
Received in revised form 30 March 2021
Accepted 17 June 2021
Available online 25 June 2021

Keywords:
Adversarial attacks
DDoS
Deep Learning
GAN
SDN

ABSTRACT

Over the last few years, Software Defined Networking (SDN) paradigm has become an emerging architecture to design future networks and to meet new application demands. SDN provides resources for improving network control and management by separating control and data plane, and the logical control is centralized in a controller. However, the centralized control logic can be an ideal target for malicious attacks, mainly Distributed Denial of Service (DDoS) attacks. Recently, Deep Learning has become a powerful technique applied in cybersecurity, and many Network Intrusion Detection (NIDS) have been proposed in recent researches. Some studies have indicated that deep neural networks are sensitive in detecting adversarial attacks. Adversarial attacks are instances with certain perturbations that cause deep neural networks to misclassify. In this paper, we proposed a detection and defense system based on Adversarial training in SDN , which uses Generative Adversarial Network (GAN) framework for detecting DDoS attacks and applies adversarial training to make the system less sensitive to adversarial attacks. The proposed system includes well-defined modules that enable continuous traffic monitoring using IP flow analysis, enabling the anomaly detection system to act in near-real-time. We conducted the experiments on two distinct scenarios, with emulated data and the public dataset CICDDoS 2019. Experimental results demonstrated that the system efficiently detected up-to-date common types of DDoS attacks compared to other approaches.

1. Introduction

Nowadays, computer network systems have become complex structures for management and control. The main reason is the number of heterogeneous devices that make up the network. The Software Defined Networking paradigm (SDN) introduced tools to simplify configuration and management, in addition to enabling more significant innovation in communication networks. The SDN paradigm enables centralized network management. Where the network control is dissociated from the data forwarding plane [1,2]. In the SDN architecture, the control plane is enhanced by centralized network control. The centralization of the control plane provides a global view of the network topology. Moreover, it enables the network traffic flow manipulation on runtime through an open and well-defined software interface [3,4]. However, the centralized control plane can be a point of vulnerability, mainly targets of Distributed Denial of Service (DDoS) attacks [5–8], which is easily flooded by many malicious requisitions. As a result, the controller may become unavailable to process normal user requisitions.

Security mechanisms are applied for detecting and preventing network systems from the actions of malicious agents. For this purpose, Network Intrusion Detection System (NIDS) is a widely used technique against Internet-based attacks [9,10]. NIDS provides a set of tools able to recognize abnormal network behaviors automatically. A NIDS can be implemented as signature-based, anomaly-based, or hybrid. In signature-based NIDS, a database of known attack pattern signatures is used to recognize intrusions. This means, an intrusion is detected when there is a match between the stored patterns and the current behavior of network activity. Anomaly-based approach focuses in generating a normal behavior profile-based historical network data. When the predicted behavior and the current behavior diverge from one another, an anomaly is detected. The main advantage of this approach is the detection of zero-day and unknown attacks. Consequently, different techniques have been used to detect anomalies [11]; recently, NIDS has been proposed by applying deep learning techniques [12,13].

Deep learning (DL) is a branch of Machine Learning (ML), where its application has become a trend, mainly due to the

* Corresponding author.
E-mail address: jlloret@dcom.upv.es (J. Lloret).

https://doi.org/10.1016/j.future.2021.06.047
0167-739X/

abstraction and generalization capacity in the learning process in many domains [14,15]. DL algorithms provide deep architectures formed by multiple layers of processing units to extract high-level abstraction from raw data. In literature, DL models such Convolution Neural Network (CNN) [16], Deep Boltzmann Machine (DBM) [17], Long Short-Term Memory (LSTM) [18], Recurrent Neural Network (RNN) [19], and Stacked Autoencoder (SAE) [20] have been applied in many areas [21–24], it includes audio processing [25], autonomous systems [26], cybersecurity [13], image and video recognition [27], and natural language processing [28]. In these fields, DL algorithms have shown their outperformance over classical ML algorithms related to classification tasks, dimensionality reduction, and feature learning algorithms.

Regarding security and management issues, deep learning technology can easily extract and learn characteristics and patterns from the network's behavior, providing useful insights for attack detection. Besides, with the fast increase in the number of users, network traffic becomes complex, and deep neural networks are powerful in reducing the network traffic complexity analysis due to their ability to learn massively complex data representations, and to handle it without human efforts [29].

Despite the extensive applications of DL algorithms in NIDS, recent studies have claimed that deep neural networks are sensitive to adversarial examples [30–32], which are performed by malicious agents to mislead DL algorithms in detecting attacks. Adversarial examples are instances with feature perturbations that are intended to cause a deep neural network to misclassify it. According to X. Zhang et al. [33] adversarial training can be applied to protect the system against adversarial example threats. In practical terms, this means adding adversarial data examples into the original dataset and using it in the model training phase. Generative Adversarial Network (GAN) framework enables to generate adversarial examples by adversarial training [34]. This GAN property can improve the NIDS performance in detecting adversarial attacks. By training the generator and the discriminator simultaneously in an adversarial way, the discriminator improves the detection rates using the generated adversarial examples and updates itself against them.

Given the significance of this issue, there is a need to ensure the security of network systems. The use of tools to support management and security activities is essential. Also, these tools must operate in an automated manner to facilitate identifying the occurrence of anomalous events and take countermeasures to minimize the effects caused by malicious agents. In this way, we present a novel anomaly detection system based on adversarial training by applying Generative Adversarial Network (GAN) for detecting and defending against up-to-date DDoS attacks.

The main contributions of this paper are listed as it follows:

- This work proposes a detection and defense system against adversarial DDoS attacks through an Adversarial Deep Learning approach, which provides a more accurate detection rate and less sensitive to adversarial examples;
- We conduct the experiments on two scenarios through up-to-date common DDoS attacks, such as NTP, DNS, LDAP, MSSQL, NetBIOS, SNMP, SSDP, UDP, UDP-Lag, WebDDoS (ARME), SYN e TFTP;
- The proposed system collects and analyzes the network traffic every second, enabling the anomaly detection system to act in near-real-time, that means, response time in up to 1 s;
- Compares the efficiency of the proposed system with different deep learning methods present in literature for detecting DDoS in SDN.

The remainder of this paper is organized as follows: Section 2 presents the related works; Section 3 introduces concepts and the structure of the proposed system; Section 4 presents the experimental scenarios and discusses the results achieved; Finally, Section 5 presents the conclusion and the future work.

2. Related work

Li et al. [35] proposed an adversarial-example attack method to mislead the in-cloud firewall-equipped Android. The authors applied a variant of GANs, called bi-objective GAN, to generate adversarial examples. The bi-objective GAN has two Discriminators. The first one attempts to distinguish malicious examples from benign examples, and the second tries to distinguish adversarial examples from normal examples. In the experiments, it was defined benign apps come from Tecncet Myapp and AndroZoo, and the malicious apps are from VirusShare. According to experimental results, over 95% of the adversarial examples generated were detected as benign by the firewall.

Zhang et al. [33] applied a Monte Carlo tree search algorithm (MCTS) to generate adversarial examples of cross-site scripting (XSS) attacks. The authors also used a GAN framework to improve the intrusion detection model to protect against adversarial attacks. In the experimental phase, to generate new types of XSS attacks, the CICIDS 2017 is used. It was extracted XSS attack traffic examples and normal traffic examples from the dataset. The GAN detection model reached a precision rate above 99% to detect XSS attacks and its adversarial examples.

Grosse et al. [36] used the adversarial training technique for malware detection against adversarial examples. First, a Deep Neural Network (DNN) model was trained on the original dataset, composed of benign and malicious applications. After that, it was applied crafted adversarial examples. With those new samples, the DDN model was retrained to improve the model's generalization, and also it makes the DNN less sensitive to adversarial examples. The dataset used to evaluate the proposed model was the DREBIN dataset.

Distributed Denial of Service is the most common attack performed against cloud computing environments [37,38]. Some solutions have been developed to address this challenge [39,40]. Velliangiri and Pandey [41] proposed an approach that combines fuzzy and taylor-elephant herd optimization (FT-EHO) inspired by the Deep Belief Network (DBN) for detecting DDoS attacks. The proposed approach taylor-elephant herd optimization has three modules: feature extraction, feature selection, and classification. The first module extracts features from raw packets, and the feature selection module selects the best features by applying the Holoentropy method. Finally, the classification module detects the DDoS attacks using FT-EHO. Kushwah and Ranga [42] also proposed a system to detecting DDoS attacks in cloud computing environments. The proposed system is based on a voting extreme learning machine (V-ELM), in which the majority results of artificial neural networks' outputs is used to get the final decision. According to the experiments' results, several ELMs can increase detection accuracy and reduce false alarms.

Akcay et al. [43] introduced a novel anomaly detection approach for images, using a conditional generative adversarial network. The proposed approach can learn the generation of high-dimensional image space and the inference of latent space. In the generator was applied an encoder–decoder–encoder to map the input image to a lower dimension vector. The generated image is mapped to its latent representation by applying an additional encoder network. The proposed approach learns the normal samples' data distribution, minimizing the distance between generated images and the latent vectors. An anomaly behavior is detected if a sample deviates a threshold from the distribution learned.

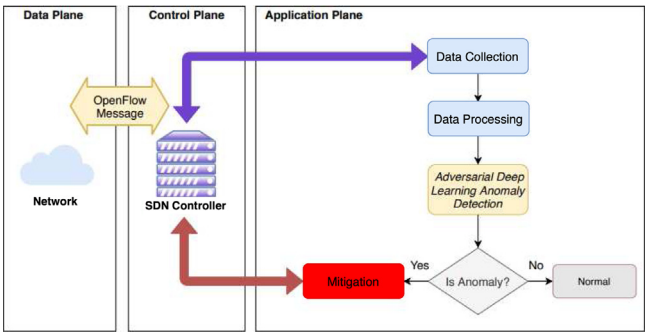


Fig. 1. Proposed system architecture using Generative Adversarial Network (GAN) framework.

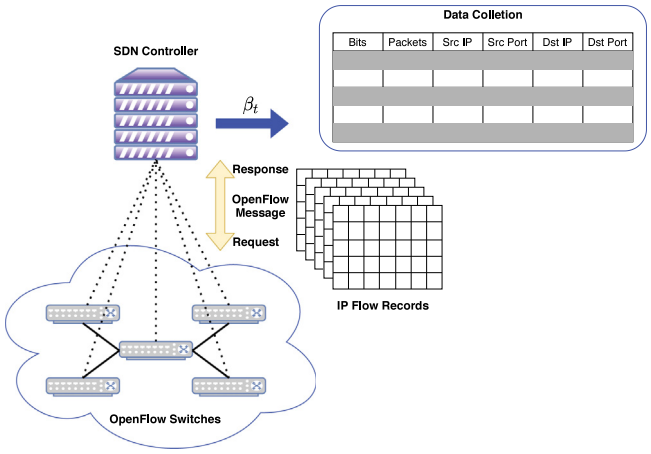


Fig. 2. Collecting Flow-Features from OpenFlow Switches.

As mentioned before, Deep Learning algorithms have shown enhanced results for detecting normal DDoS attacks. However, few models are designed for detecting adversarial examples of DDoS attacks. The adversarial DDoS attacks refer to the type of violation to carry out a network attack causing misclassifications, where an attack is classified as normal (false negative). Undetected DDoS attacks can inflict damage on network resource availability [44]. For this purpose, we designed a system that applies adversarial training to overcome this vulnerability and improve the anomaly detection event rates in SDN.

3. Proposed system

According to previous studies [45,46], the DDoS attacks are commonly carried out by malicious agents against network systems. Centralization of network control on a controller in the SDN architecture becomes a vulnerability exploited by DDoS attacks. In this regard, we proposed a system that acts in the application plane to detect and mitigate this threat. In Fig. 1, a flowchart depicts the proposed system architecture. The system can be summarized in the following steps:

1. Data Collection: Every second, the controller collects IP flows from the tables flows of the switches that belong to the network.
2. Data Processing: In this step, categorical IP flows features (e.g., destination/ source IP addresses and ports numbers) are casting in numerical features.
3. Adversarial Deep Learning Anomaly Detection: The module analyzes the network's behavior and detects the occurrence of DDoS attacks.
4. Mitigation: In case a DDoS is detected, the mitigation module takes countermeasure to minimize the damages.

3.1. Data collection

Data collection is a fundamental step for continuous monitoring of network activities. Besides, it is a prerequisite for detecting possible anomalies. For this purpose, the proposed system contains a module for acquisition of traffic data. In that module, the network information is collected from the switches using the OpenFlow protocol. The protocol uses the flow-based concept to identify network traffic, in which the traffic is handled in terms of flows rather than individual packets. As defined in [47], the flows are a packets sequence passing an observation point in the network during a specific time interval. All packets in a flow have common properties such as transport protocol, IP addresses, and ports from both source and destination.

Previous works in the literature collected information from the network data using time intervals between 1 and 5 min [48,49].

Table 1
Collected flow features.

Flow Feature	Description
$x^{bits}_{\in\alpha_i}$	The amount of bits belongs to the flow α_i
$x^{packets}_{\in\alpha_i}$	The amount of packets belongs to the flow α_i
$x^{srcIP}_{\in\alpha_i}$	Source IP address
$x^{srcPort}_{\in\alpha_i}$	Source port number
$x^{dstIP}_{\in\alpha_i}$	Destination IP address
$x^{dstPort}_{\in\alpha_i}$	Destination port number

However, this analysis interval has been reduced due to the high transmission rates reached in the current network scenario. The new solutions present in the literature have used intervals analysis near-real-time [50,51]. In that way, we reduce the time interval analysis to one-second. Every second, the proposed system requests and analyzes the IP flow records collected from each switch. This interval enables the system to react fast against anomaly events by reducing the time response.

For all time t, the controller sends requests to collect IP flow records for switches belonging to the network through OpenFlow's Read-State messages. After each switch receives this message, they send a response containing each flow record stored in their forwarding table. This process is represented in Fig. 2. The collected data can be defined as a set $\beta_t = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, where each α_i is a flow record that is composed by the following features presented in Table 1:

The collected flow features can be classified as quantitative (bits and packets) and qualitative (source IP address, destination IP address, source, and destination ports). The quantitative flow features provide information about traffic volume, which is essential to understand the amount of traffic transported on the network. On the other hand, the qualitative flow features allow us to understand which devices communicate and which applications are being accessed.

3.2. Data processing

The previous module's IP flow records need to be processed to extract specific characteristics that assist in the anomaly detection approach. The data processing module is responsible for grouping the flow attributes in each analysis interval. Every second, the following attributes are processed:

- *bits/s*
- *packets/s*

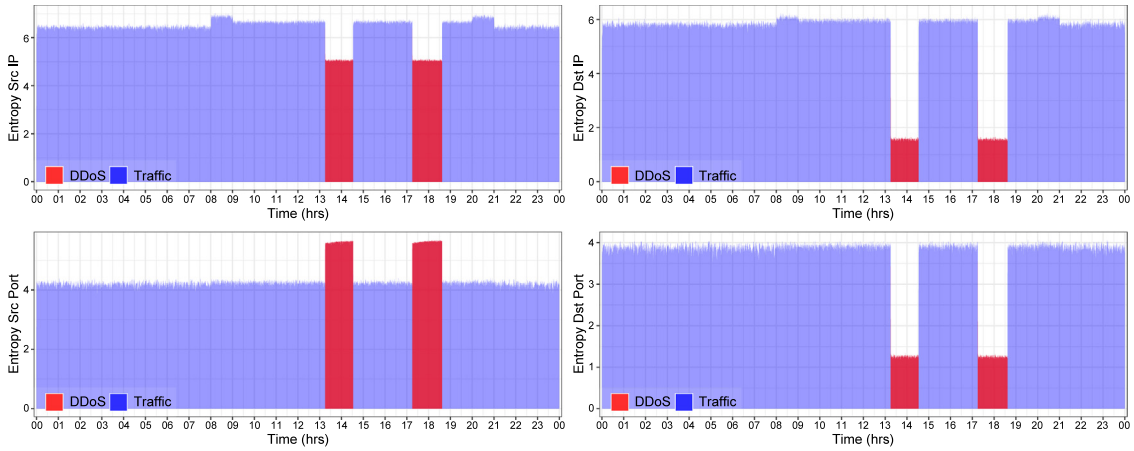


Fig. 3. Flow features entropy analysis during DDoS attacks.

- *Source IP Entropy*
- *Source Port Entropy*
- *Destination IP Entropy*
- *Destination Port Entropy*

These flow features were extensively analyzed and employed in previous works present in the literature [52]. They have been used in network traffic characterization of high speed, and the outcomes achieved for anomaly detection have been effective.

The rate of bits and packets per second, the quantitative features, are processed by:

$$bits/s = \sum_{j=1}^{|\beta_t|} x_j^{bits} \tag{1}$$

$$packets/s = \sum_{j=1}^{|\beta_t|} x_j^{packets} \tag{2}$$

IP addresses and port numbers are categorical features. However, the detection module applies a deep neural network to detect anomalies events. The deep neural approaches require that input variables are numbers. Thus, the qualitative features, IP and ports, must be casting to numerical type. A simple transformation is to group these data by calculating the entropy. In this module, we applied the Shannon Entropy [53], which highlights the concentration or dispersion degree of the traffic features during an analyzed time interval. Given a set of a qualitative feature, such as $x^{srcIP} = \{x_1, x_2, \ldots, x_n\}$, in which a x_i represents the occurrences number of source IP address i at a given time interval t. The Shannon Entropy $H(\cdot)$ for the flow feature x^{srcIP} is defined as:

$$H(x^{srcIP}) = -\sum_{i=1}^{n} \left(\frac{x_i}{S}\right) \log_2 \left(\frac{x_i}{S}\right), \tag{3}$$

in which $S = \sum_{i=1}^{n} x_i$ is the sum of all occurrences of the elements present on the analyzed time interval t. The entropy calculation for the other qualitative attributes (like $H(x^{srcPort})$, $H(x^{dstIP})$, and $H(x^{dstPort})$) is calculated in the same manner as presented in the Eq. (3).

Information on the degree of concentration or dispersion of a given flow feature can help during the anomaly detection phase. For instance, under a DDoS attack occurrence, the destination IP address and destination Port number distribution may become concentrated due to the high number of connections requested by the attackers. The concentration and dispersion of the flow attributes affected during the occurrence of a DDoS attack is illustrated in Fig. 3. This figure presents the analysis results of Source IP Entropy, Destination IP Entropy, Source Port Entropy, and Destination Port Entropy for a day of network traffic used to evaluate our approach. The blue intervals represent the entropy measured during the normal behavior of the network. As can be seen, the entropy value remains continuous, without significant variations in the normal ranges. On the contrary, the intervals in red represent the occurrence of DDoS attacks. There is a concentration in these intervals of the source and destination IP addresses and the destination port, as shown. On the other hand, there is a dispersion of the source port's entropy because multiple attackers use random source ports.

3.3. Adversarial Deep Learning Anomaly Detection

The Adversarial Deep Learning Anomaly Detection step aims to detect and identify malicious agents' activities that can cause anomalous behavior. In this module, it is necessary to use techniques capable of differentiating the normal traffic operation from possible attacks against the network. Recently deep neural networks (DNN) have been applied in anomaly detection systems [9, 13]. These systems have outperformed all the other classical machine learning systems. Though DNN approaches suffer against adversarial examples. To address this problem, we apply adversarial training through Generative Adversarial Network (GAN) framework.

GAN is a framework proposed by Goodfellow et al. [34], which simultaneously trains two models through an adversarial process. The GAN framework comprises two neural networks models, a generative model (G) and a discriminative model (D). Both models compete against each other in an adversarial way. Considering that, this competition consists of a minimax two-player game according to the game theory scenario. The G model generates fakes samples from a noise distribution similar to the original dataset. On the other hand, the D model determines the probability that the samples belong to the original dataset.

The G model builds a mapping from a prior noise p_z to a data space $G(z)$ and learns a generative distribution p_g over the data x. In order to increase the error rate of D, the fake samples must be as similar as possible to the real distribution p_{data}. The Eq. (4) represents the objective function of G model:

$$\min \frac{1}{2} \mathbb{E}_{z \sim p_z} \log(1 - D(G(z))) \tag{4}$$

where $z \sim p_z$ is data from the distribution generated by G and $D(G(z))$ indicates the probability of D determining the data generated by G. The G model achieves its training minimizing Eq. (4), which means that the D model predicts the generated data $G(z)$

Table 2
Structure of the generator (G) model.

#	Layer	Neurons	Activation
1	Fully-connected (Dense)	6	ReLU
2	Fully-connected (Dense)	10	ReLU
3	Fully-connected (Dense)	8	ReLU
4	Fully connected (Dense)	6	Linear

Table 3
Structure of the discriminator (D) model.

#	Layer	Neurons	Activation
1	Fully-connected (Dense)	12	ReLU
2	Fully-connected (Dense)	10	ReLU
3	Fully-connected (Dense)	6	ReLU
4	Fully connected (Dense)	1	Sigmoid

with a high probability. The D model aims to classify whether a sample is a real data or a generated data. Thus, the objective function of D is defined in Eq. (5):

$$Max \frac{1}{2}\mathbb{E}_{x\sim pdata} \log D(x) + \frac{1}{2}\mathbb{E}_z \log(1 - D(G(z))) \quad (5)$$

where $D(x)$ determines the probability of the real data and $x\sim p_{data}$ is the data from the original distribution. The D model effectiveness is achieved maximizing the Eq. (5). So that, considering the conflict between the two models, the GAN framework can be defined as minimax game. Both models continuously improve their effectiveness until an equilibrium is reached. Eq. (6) formalized the minimax game:

$$minMax \frac{1}{2}\mathbb{E}_{x\sim pdata} \log D(x) + \frac{1}{2}\mathbb{E}_z \log(1 - D(G(z))) \quad (6)$$

As previously identified, the GAN framework includes the generator model (G) and the discriminator model (D). Both models implemented in our approach are Deep Neural Networks (DNNs) structure. Neural networks with multiple hidden layers are qualified as Deep Neural Networks [36]. DNNs enable hierarchically extracting knowledge abstraction and learning characteristics and patterns from the raw network data. Besides, DNNs give more representation of input data to improve the rate detection of complex attacks in a high-speed network environment, mainly DDoS attacks [54,55].

The generator (G) was implemented with multiple fully-connected layers (Dense) and a linear output layer. Table 2 shows the G model structure. We also implemented multiple fully-connected layers for the discriminator (D), but we implemented a sigmoid layer for the output layer, in which output is the classification of the network's behavior in that analysis interval. In that manner, our approach classifies the network traffic as normal or DDoS attack. Besides, the discriminator (D) during the adversarial training learns the network's normal behavior, which is advantageous for detecting zero-day attacks. Table 3 details the D model's layers.

The flowchart present in Fig. 4 illustrated the adversarial training process used in our approach. The first step is training the D model for d_{epoch}. During this step, a minibatch of m samples from the training dataset are randomly sampled. The training dataset is composed of normal traffic and DDoS attacks. Then, the G model generates adversarial a examples from z random noise. Both m and a sets are used to training the D model. The weights of the D and G models are updated according to its stochastic gradient. The stochastic gradients of D and G models are defined in Eqs. (7) and (8), respectively.

$$\nabla_{\theta_D} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(m^{(i)}\right) + \log\left(1 - D(G(z^{(i)}))\right) \right] \quad (7)$$

$$\nabla_{\theta_G} \frac{1}{z} \sum_{i=1}^{z} \log\left(1 - D(G(z^{(i)}))\right) \quad (8)$$

The adversarial training process of generating and training is repeated until the D model can detect the fake samples or a number max of training iterations t. This process is illustrated as follows in Algorithm 1:

Algorithm 1 GAN Adversarial training steps

Require: Generator model G; Discriminator model D; Training dataset

while t iterations training or stop condition not met **do**
  for d_{epoch} **do**
    Sample minibatch of m from the training dataset
    Generates a adversarial examples from G
    Update the discriminator by ascending its stochastic gradient by Eq. (7)
  end for
  Sample minibatch of z samples from noise prior $z\sim p_z$
  Update the generator by descending its stochastic gradient by Eq. (8)
end while

return G and D;

3.4. Mitigation

The mitigation module is triggered after a DDoS attack is detected by the Adversarial Deep Learning Anomaly Detection. This module aims to take countermeasures to minimize the damages caused during an attack. In previous work [56], we have proposed a mitigation anomaly approach that achieved efficient outcomes. In this manner, we extended the proposed mitigation approach to this module.

Basically, the approach described in [56] is **Event-Condition-Action** (ECA) model-based, in which the **Event** refers to a specific anomaly associated with a set of rules. The Condition describes the rules where a specific anomaly event took place. Lastly, the **Action** is a countermeasure taking against an anomaly event.

Some network anomalies (e.g., Flash crowd) have characteristics similar to an attack DDoS attack, but they are requests from legitimate users. To overcome this vulnerability, a *Safe List* mechanism was implemented, which maintains a list of flow features from legitimate users.

The mitigation process is summarized as follow in Algorithm 2.

Algorithm 2 Mitigation Process.

Require: Suspect flows β_t

Identify the suspect flows based on IP addresses and ports that make the analysis interval anomalous
Identify the destination IP address which receives the most flows
Identify in those flows the attackers' IP address which have the same destination port
if IPs e ports are on the *Safe List* **then**
  Forward packets
else
  Drop packets
end if

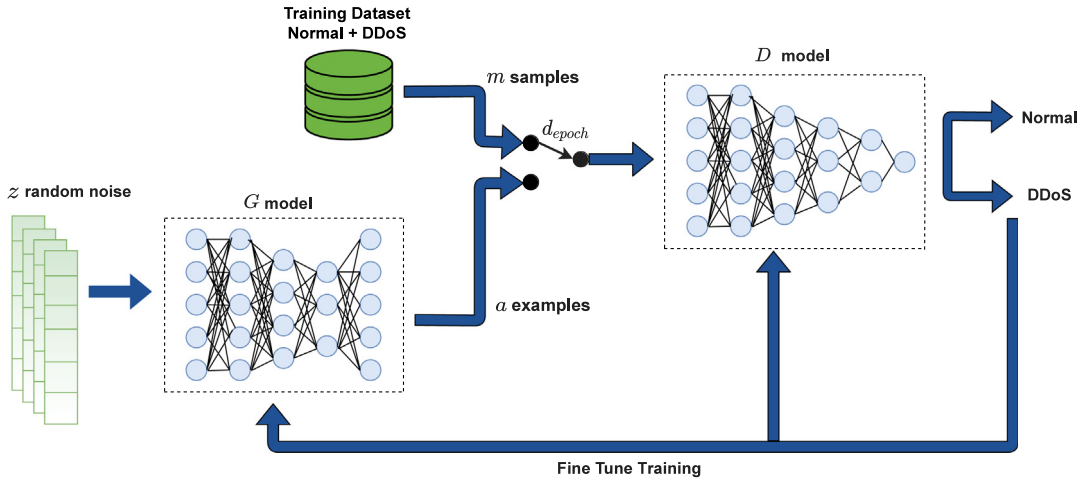


Fig. 4. GAN training structure.

4. Experimental results and discussion

To evaluate the performance of the proposed system, we applied tests in different scenarios and compared it with some other systems present in literature. We used the Python language with development libraries for Deep Learning application Keras [57] and TensorFlow [58]. During the training step the parameters were set as, Dropout with rate of 0.2 to prevent overfitting [59]; the loss function Binary Crossentropy that is a classical loss function used in binary classification tasks; the default learning rate equals to 0.001 defined in Keras [57], and optimizer was set as Adam [60], which is an adaptive learning rate optimization algorithm for training deep neural networks. The experiments were made in an environment with the following configuration: Intel Core i5 2.21 GHz, 8 GB RAM, Windows 10 system.

The first test scenario contains data from the emulation of an SDN network with a high transmission rate and many devices connected, totaling 128 hosts. In this scenario, it has two periods of UDP DDoS attacks with different intensities. In the second scenario, we applied the public dataset CICDDoS 2019 [61] from the Canadian Institute for Cybersecurity. This dataset contains 28 normal profile network behavior and up-to-date types of DDoS attacks.

4.1. Evaluation metric

To analyze the proposed system's performance, we adopted the following classic metrics: Accuracy, Precision, Recall, and F1 Score. Accuracy metric indicates the percentage of intervals correctly classified. Precision presents the percentage of intervals classified as DDoS, which are DDoS. Recall shows how effective the model is in identifying DDoS intervals related to all the intervals. The definition of these metrics can be given by:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \times 100\% \tag{9}$$

$$Precision = \frac{TP}{TP + FP} \times 100\% \tag{10}$$

$$Recall = \frac{TP}{TP + FN} \times 100\% \tag{11}$$

where TP, TN, FP, and FN mean True Positive, True Negative, False Positive, and False Negative, respectively.

Finally, $F1$ Score is the harmonic mean between recall and precision, providing an appropriate score to the system performance. Its statistical formulation is given by Eq. (12):

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{12}$$

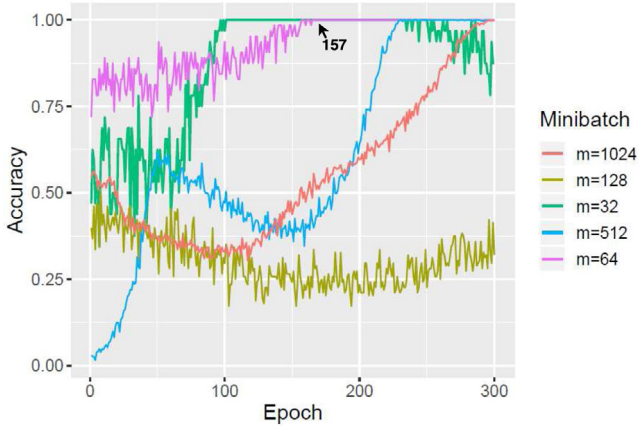


Fig. 5. Minibatch size parameter evaluation.

4.2. Parameters evaluation

During the GAN framework training phase we evaluated some parameters, such as minibatch size and the number of d_{epoch} for training the discriminator before updating the generator. These parameter were evaluated to finding a point of equilibrium between the generator and discriminator.

To evaluate the minibatch size, we used 32, 64, 128, 256, 512, and 1024 samples. The number of epochs and accuracy rate were evaluated to obtain the convergence value. Fig. 5 illustrates the results of this test. A minibatch size of 32 reached better accuracy rate first, but it decreased after 257 epochs. As observed, the best result reached was using a minibatch size of 64, in which the number of epochs required for convergence was 157.

The next parameter evaluated was the number of d_{epoch} for alternating and updating the generator and the discriminator. This test is detailed in Fig. 6. According to the information presented in [33], small values can reach better results. In that manner, the estimation of the referred parameter d_{epoch} was applied between 1 to 3. The number of epochs and accuracy rate were evaluated to obtain the appropriate value of d_{epoch}. As the results achieved in Fig. 6, the number of d_{epoch} that reached the appropriate convergence was equal to 2, in which the number of epochs required for training were 157. The other values of d_{epoch} 1 and 3 are unstable and do not converge.

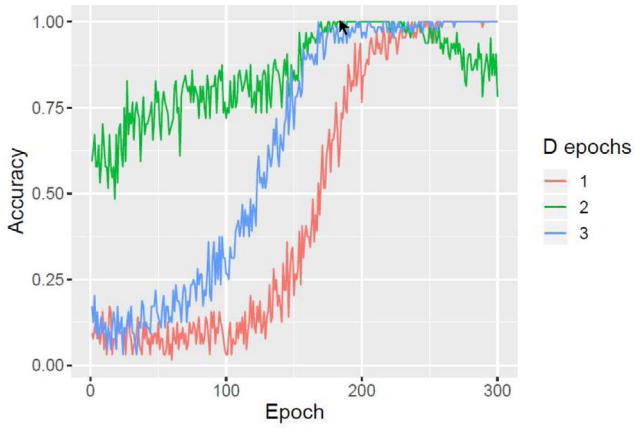


Fig. 6. Estimation of the parameter d_{epoch}.

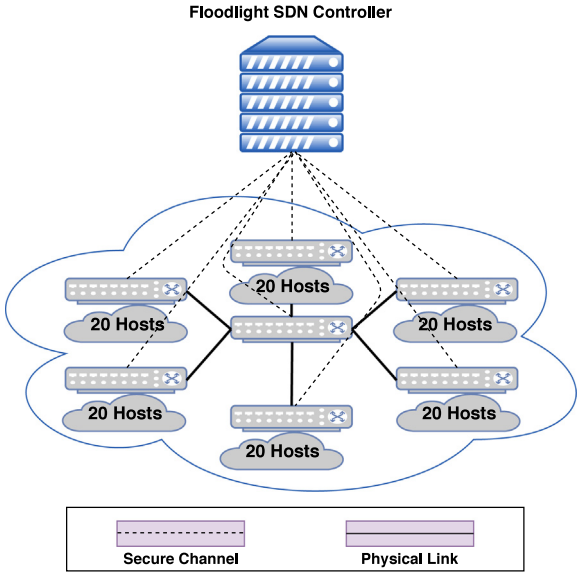


Fig. 7. Emulated SDN network on scenario one.

4.3. Scenario 1: SDN emulated environment

In this scenario, we emulated a network topology through the Mininet [62] network emulator. The Mininet emulator is widely applied in developing SDN solutions due to the facility for implementing realistic virtual SDN environments composed of controllers, hosts, links, and switches on one virtual machine. The network arrangement emulated is a star topology in which six switches are connected to a central switch. Every subnetwork contains 20 hosts, totaling 120 hosts, as described in Fig. 7. Besides, we used the SDN controller Floodlight [63], a controller based on Java that has been utilized in many SDN applications as a network controller.

We emulated the behavior of a network with high transmission rates for 24 h. To inject traffic in the emulated network, we used a tool called Scapy [64], a packet manipulation tool for computer networks. This tool provides an emulated network that can be similar to a real network scenario.

Two UDP DDoS flood attacks were carried out during the emulation stage with different intensities and duration time. This type of attack is the common DDoS method used by attackers against network services [8]. The parameters used in the attacks are shown in detail in Table 4. This dataset is available online.[1]

Table 4
UDP DDoS flood parameters.

Type of attack	Attack parameters
DDoS #1	Attackers: 15 Attacking IPs: 10.0.0.21–10.0.0.35 Victim IP address: 10.0.0.92:2000 Time: 10:10–11:20
DDoS #2	Attacking IPs: 10.0.0.45–10.0.0.60 Victim IP address: 10.0.0.33:8080 Time: 14:45–16:00

We evaluated the GAN framework performance and compared it with other neural networks-based methods present in literature that were also applied to detect DDoS attacks in SDN environments. The methods are the Convolutional Neural Network (CNN) [65], the Long Short-Term Memory (LSTM) [66]; and finally, the classical neural network Multilayer Perceptron (MLP) [54]. These methods have reached accurate outcomes in detecting attacks in SDN, and we compared our system performance with them.

Fig. 8 shows the outcomes achieved by each one of the compared methods through the evaluated metrics. Regarding the metrics, the methods GAN, CNN, LSTM, MLP presented values greater than 95%. For *Accuracy*, GAN and CNN obtained similar results, with values of 99.78 and 99.69, respectively. The LSTM reached an accuracy rate of 97.21%, and the MLP reached 95.91%, which means these last two methods were less accurate in classifying the analysis intervals.

The next evaluated metric was the *Precision* rate. GAN achieved a Precision rate of 99.76%, an improvement in 2.19%, 2.38%, and 4.26% higher than CNN, LSTM, and MLP, respectively. In the following, we evaluated the *Recall* metric. GAN framework also fared better, with a rate of 99.99% for this metric, followed by CNN, LSTM, and MLP methods, reaching rates of 99.91%, 96.90%, and 95.69%, respectively.

Finally, we further examined the robustness of all compared methods using the $F1$ score. GAN reached the best result, with a rate of 99.87%, the remaining ones were CNN, LSTM, and MLP, reaching rates of 98.73%, 97.13%, and 95.74%, respectively.

According to the results achieved for all evaluated metrics, the proposed GAN framework presented an adequate performance in detecting DDoS attacks. Fig. 9 presents a radar chart that summarizes all compared methods' performance metrics previously addressed in a single chart. The GAN is closer to the outer circle, which means that closer to 100% the analyzed method fared better for the four evaluated metrics, followed by CNN, LSTM, and MLP.

The analysis intervals detected as attacks by the detection module are reported to the mitigation module. In these intervals, mitigation policies are applied. Fig. 10 presents the behavior of the six attributes flow analyzed before and after the mitigation process. The blue bar plot and the green line plot illustrate network traffic before and after applying the mitigation policy. An increase in the bits and packets rates can be noticed before applying the mitigation. After the mitigation, the network traffic behavior tends to go back to its normality due to anomalous packets' discards.

4.4. Scenario 2: CICDDoS 2019 dataset

In this second test scenario, we evaluated the proposed system's performance for detecting different types of DDoS attacks.

[1] http://www.uel.br/grupos/orion/datasets.html

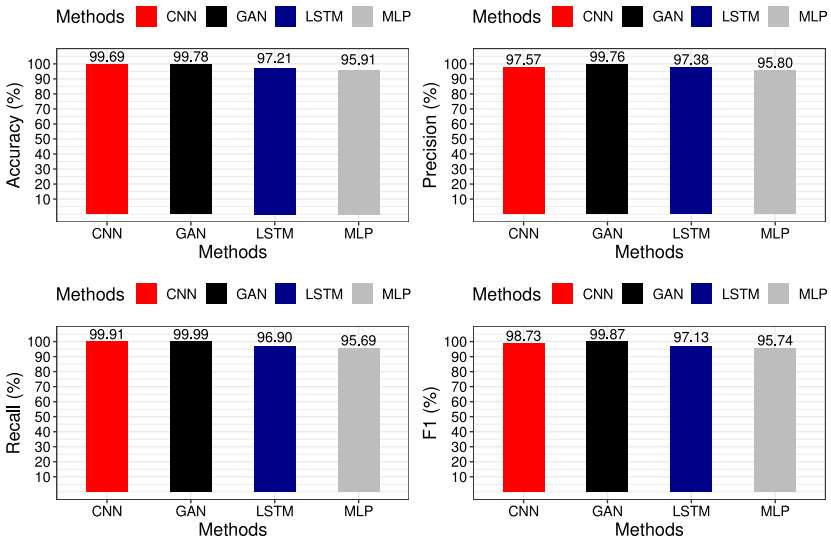


Fig. 8. Comparison outcomes between GAN and the compared methods through the evaluated metrics on first scenario.

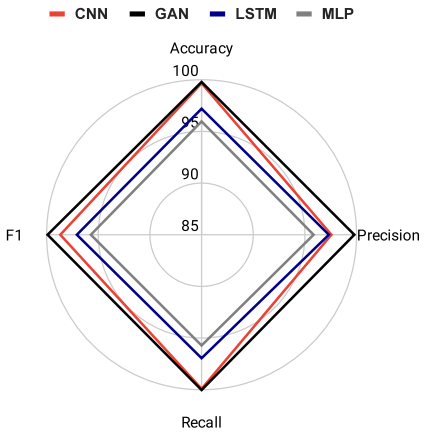


Fig. 9. Radar chart results for the evaluated methods on first scenario.

In order to do so, we applied the public dataset called CICDDoS 2019. This dataset contains 28 normal profile network behaviors and the most up-to-date common types of DDoS attacks. The dataset has been organized by day, one for training and another for testing. The training dataset comprises 12 different modern DDoS attacks such as NTP, DNS, LDAP, MSSQL, NetBIOS, SNMP, SSDP, UDP, UDP-Lag, WebDDoS (ARME), SYN, and TFTP. The testing day contains 6 types of DDoS attacks, including NetBIOS, LDAP, MSSQL, UDP, UDP-Lag, and SYN. In that manner, the DDoS types explored in this scenario are summarized as follow:

- Network Time Protocol (NTP) server functionality are used to overwhelm a targeted network by sending UDP packets with spoofed IP addresses;
- Domain Name System (DNS) is an attack that the attacker overwhelms a particular server in order to disrupt that domain which can be carried out using either TCP or UDP packets;
- Lightweight Directory Access (LDAP) is a protocol used for directory services on corporate networks. The attacker sends

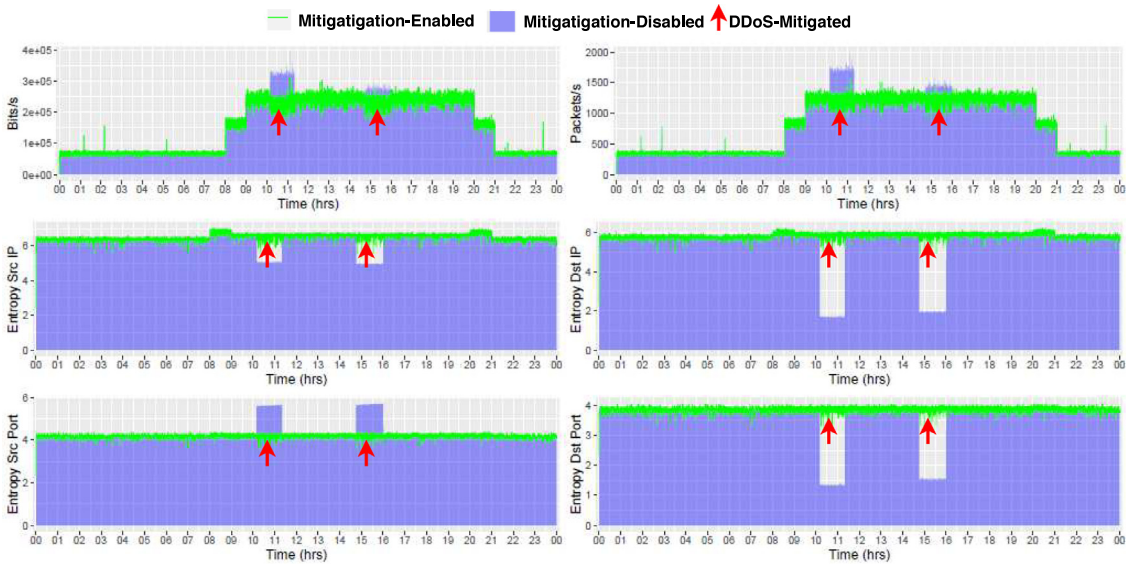


Fig. 10. Flow features behavior before and after the mitigation process on scenario one.

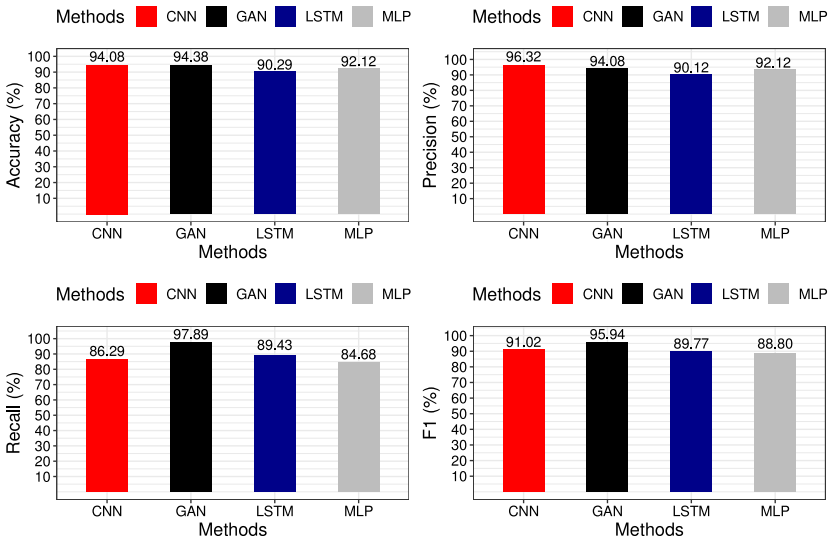


Fig. 11. Comparison outcomes between GAN and the compared methods through the evaluated metrics on second scenario.

small requests to a vulnerable LDAP server to produce amplified replies, which floods the victim;
- Microsoft Structured Query Language (MSSQL) is an attack that makes it possible to execute malicious SQL queries;
- NetBios is an attack that sends many spoofed "Name Release" or "Name Conflict" messages, forcing the machine to remove its own name from its name table and becoming unable to other NetBIOS;
- Simple Network Management Protocol (SNMP) attack is a type of DDoS in which the attacker sends a large number of requests with spoofed IP address to several devices;
- Simple Service Discovery Protocol (SSDP) attack is a type of DDoS that uses Universal Plug and Play (UPnP) networking protocols to send a massive amount of traffic to a targeted victim;
- User Datagram Protocol (UDP) packets are sent by an attacker to overwhelm the target's ability to process and respond to it;
- UDP-lag is a type of attack that interrupts the connection between the target client and the server;
- WebDDoS (ARME) attack is a DDoS to crash the website or slow it by flooding the network with multiple fake requests to the target;
- SYN attack is a DDoS in which packets with spoofed IP addresses are sent to overwhelm all available ports to a targeted server;
- Trivial File Transfer Protocol (TFTP) attack occurs when an attacker overflows the buffer server.

GAN efficiency measurements were performed using the same classical metrics as carried out in the first scenario. We also compared the results achieved with the methods CNN [65], LSTM [66], and MLP [54]. Fig. 11 illustrates the measurement results of the metrics obtained for each of them.

As it can be noticed from Fig. 11, the methods GAN and CNN reached similar results for the *Accuracy* metric, with rates of around 94% (GAN reached 0.3% greater than CNN). The other two methods, LSTM and MLP, reached Accuracy rates of 90.29% and 92.12%, respectively.

Regarding the *Precision* metric, the CNN method fared relatively better than the other compared methods, reaching a rate of 96.32%, followed by GAN, MLP, and LSTM with rates of 94.08%, 92.12%, and 90.12%, respectively. For the *Recall* metric, the GAN obtained superior performance, achieving a rate of 97.89% for this

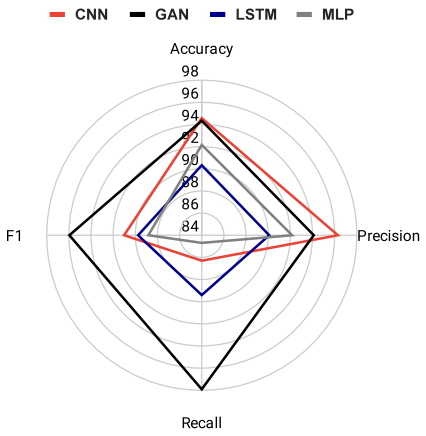


Fig. 12. Radar chart results for the evaluated methods on second scenario.

metric, a rate improvement of 8.46%, 11.6%, and 13.21% higher than LSTM, CNN, and MLP, respectively.

As mentioned in the previous scenario, we also used the F1 score to determine how precise and robust the methods are in detecting different types of DDoS attacks. Regarding the $F1$ score, it is clear that the GAN framework yielded better outcomes than the other methods, with a rate of 95.54%. The methods CNN, LSTM, MLP fared similar rates, with 91.02%, 89.77%, and 88.80%, respectively.

We summarized all outcomes measurements achieved by the tested methods in a radar chart. These outcomes are illustrated in Fig. 12. The methods compared in this scenario obtained significantly lower results compared to the first scenario. The main reason for this difference is the number of attacks evaluated, making the detection process difficult. However, GAN achieved superior results due to its training process that used an adversary training approach, making it less sensitive to different attacks. As the results gained in the first scenario, the GAN framework also reached the best outcomes on average. The proposed system provides an efficient approach capable of detecting different kinds of DDoS attacks.

In this scenario, we also evaluated the mitigation performance against several DDoS attacks. The anomaly detection module triggered the anomalous intervals to the mitigation module, in

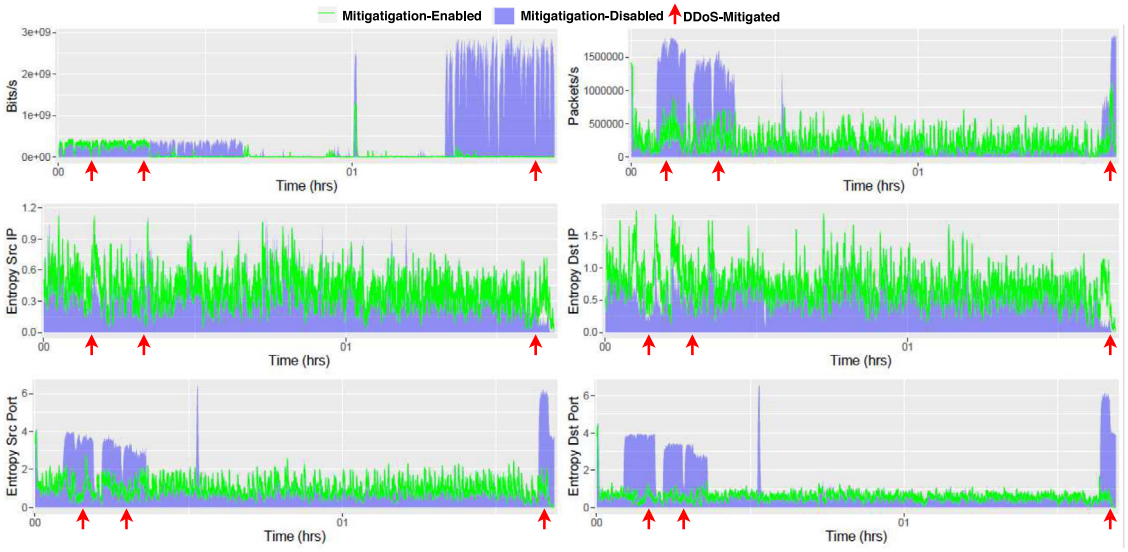


Fig. 13. Flow features behavior before and after the mitigation process on CICDDoS 2019 dataset.

which the module automatically applied the DDoS countermeasure. Fig. 13 shows the network traffic behavior for each flow feature. The traffic before the mitigation is represented in the blue area, and the green line shows the traffic after applying the mitigation policy. As it can be seen in Fig. 13 the flow features variations were inhibited after the mitigation module is triggered. Through this analysis, the mitigation module could mitigate the DDoS attacks successfully, and the normal traffic has been maintained.

5. Conclusion

In this work, we proposed a system detection and defense against DDoS attacks in SDN environments. This system comprises four integrated modules that provide tools for collecting, processing, detecting, and mitigating attacks. We used the Generative Adversarial Network (GAN) and applied adversarial training to make the system less sensitive to adversarial attacks.

We compared the proposed system's outcomes with different neural network methods present in literature for detecting DDoS in SDN, such as CNN, LSTM, and MLP. During the test stage, the methods were submitted to two test scenarios. The first scenario emulated a real SDN environment with many hosts and high transmission rates, using the Mininet emulator and the Floodlight controller. In the second scenario, we evaluated the system through the recent public dataset CICDDoS 2019, which contains the most 12 up-to-date common types of DDoS attacks such as NTP, DNS, LDAP, MSSQL, NetBIOS, SNMP, SSDP, UDP, UDP-Lag, WebDDoS (ARME), SYN, and TFTP.

In the first scenario, we evaluated the proposed system for detecting UDP flood attacks in an SDN network environment with high transmission rates. The network traffic was analyzed near-real-time, which was performed in one-second time intervals. The results gained by the prosed system have fared better than the other compared methods in that test environment. According to the evaluated metrics, the system was able to detect and defend against almost DDoS performed.

In the second scenario, we tested the system performance for detecting DDoS attacks against different applications. In that scenario, the compared methods were less accurate in detecting these attacks. On the other side, we optimized our system using the GAN framework through adversarial training that provided adversarial examples to enhance its ability to defend against them, making the system more accurate in identifying the different DDoS attacks. In both scenarios, the proposed system obtained results superior to the other compared methods, allowing its application against several DDoS attacks on SDN environment.

The GAN framework has shown its potential for use in environments susceptible to different threats due to its adversarial training that makes the system less sensitive to adversarial attacks. In conclusion, the GAN framework improved the performance indicators evaluated in this work and showed the powerful capacity to detect novel attacks.

In our future work, we intend to explore the impact of other deep neural methods (e.g., Gated-Recurrent Unit, Stacked Auto-Encoder) on the General and Discriminator. We can consider applying tests, increasing the number of hosts and switches on the emulated scenario. Besides, we will pay efforts to detect and classify other attacks in a multiclass classification approach.

CRediT authorship contribution statement

Matheus P. Novaes: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing - original draft, Visualization, Funding acquisition. **Luiz F. Carvalho:** Data curation, Writing - review & editing. **Jaime Lloret:** Supervision, Writing - review & editing. **Mario Lemes Proença Jr.:** Conceptualization, Formal analysis, Writing - review & editing, Supervision, Project administration, Investigation, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been partially supported by the National Council for Scientific and Technological Development (CNPq) of Brazil under Grant of Project 310668/2019-0 and by SETI, Brazil/Fundação Araucária due to the concession of scholarships; by the "Ministerio de Economía y Competitividad, Spain" in the "Programa Estatal de Fomento de la Investigación Científica y Técnica de Excelencia, Subprograma Estatal de Generación de Conocimiento" within the project under Grant TIN2017-84802-C2-1-P.

References

[1] X. Zhang, L. Cui, K. Wei, F.P. Tso, Y. Ji, W. Jia, A survey on stateful data plane in software defined networks, Comput. Netw. (2020) 107597, http://dx.doi.org/10.1016/j.comnet.2020.107597, URL http://www.sciencedirect.com/science/article/pii/S1389128620312305.

[2] T. Das, V. Sridharan, M. Gurusamy, A survey on controller placement in SDN, IEEE Commun. Surv. Tutor. 22 (1) (2020) 472–503, http://dx.doi.org/10.1109/COMST.2019.2935453.

[3] O. Salman, I. Elhajj, A. Chehab, A. Kayssi, IoT survey: An SDN and fog computing perspective, Comput. Netw. 143 (2018) 221–246, http://dx.doi.org/10.1016/j.comnet.2018.07.020, URL http://www.sciencedirect.com/science/article/pii/S1389128618305395.

[4] S. Garg, K. Kaur, N. Kumar, J.J.P.C. Rodrigues, Hybrid deep-learning-based anomaly detection scheme for suspicious flow detection in SDN: A social multimedia perspective, IEEE Trans. Multimed. 21 (3) (2019) 566–578, http://dx.doi.org/10.1109/TMM.2019.2893549.

[5] C. Gkountis, M. Taha, J. Lloret, G. Kambourakis, Lightweight algorithm for protecting SDN controller against DDoS attacks, in: 2017 10th IFIP Wireless and Mobile Networking Conference, WMNC, 2017, pp. 1–6, http://dx.doi.org/10.1109/WMNC.2017.8248858.

[6] K.S. Sahoo, D. Puthal, M. Tiwary, J.J.P.C. Rodrigues, B. Sahoo, R. Dash, An early detection of low rate DDoS attack to SDN based data center networks using information distance metrics, Future Gener. Comput. Syst. 89 (2018) 685–697, http://dx.doi.org/10.1016/j.future.2018.07.017, URL https://www.sciencedirect.com/science/article/pii/S0167739X18309543.

[7] M.P. Singh, A. Bhandari, New-flow based DDoS attacks in SDN: Taxonomy, rationales, and research challenges, Comput. Commun. 154 (2020) 509–527, http://dx.doi.org/10.1016/j.comcom.2020.02.085.

[8] O. Yurekten, M. Demirci, SDN-based cyber defense: A survey, Future Gener. Comput. Syst. 115 (2021) 126–149, http://dx.doi.org/10.1016/j.future.2020.09.006, URL http://www.sciencedirect.com/science/article/pii/S0167739X20303277.

[9] Ömer KASIM, An efficient and robust deep learning based network anomaly detection against distributed denial of service attacks, Comput. Netw. 180 (2020) 107390, http://dx.doi.org/10.1016/j.comnet.2020.107390, URL http://www.sciencedirect.com/science/article/pii/S1389128620304114.

[10] N. Garcia, T. Alcaniz, A. González-Vidal, J.B. Bernabe, D. Rivera, A. Skarmeta, Distributed real-time slowdos attacks detection over encrypted traffic using artificial intelligence, J. Netw. Comput. Appl. 173 (2021) 102871, http://dx.doi.org/10.1016/j.jnca.2020.102871, URL http://www.sciencedirect.com/science/article/pii/S1084804520303362.

[11] N. Moustafa, J. Hu, J. Slay, A holistic review of network anomaly detection systems: A comprehensive survey, J. Netw. Comput. Appl. 128 (2019) 33–55, http://dx.doi.org/10.1016/j.jnca.2018.12.006.

[12] A. Aldweesh, A. Derhab, A.Z. Emam, Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues, Knowl.-Based Syst. 189 (2020) 105124, http://dx.doi.org/10.1016/j.knosys.2019.105124, URL http://www.sciencedirect.com/science/article/pii/S0950705119304897.

[13] M.A. Ferrag, L. Maglaras, S. Moschoyiannis, H. Janicke, Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study, J. Inform. Secur. Appl. 50 (2020) 102419, http://dx.doi.org/10.1016/j.jisa.2019.102419.

[14] D.-T. Hoang, H.-J. Kang, A survey on deep learning based bearing fault diagnosis, Neurocomputing 335 (2019) 327–335, http://dx.doi.org/10.1016/j.neucom.2018.06.078, URL http://www.sciencedirect.com/science/article/pii/S0925231218312657.

[15] A.M. Ozbayoglu, M.U. Gudelek, O.B. Sezer, Deep learning for financial applications : A survey, Appl. Soft Comput. 93 (2020) 106384, http://dx.doi.org/10.1016/j.asoc.2020.106384, URL http://www.sciencedirect.com/science/article/pii/S1568494620303240.

[16] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.

[17] D.H. Ackley, G.E. Hinton, T.J. Sejnowski, A learning algorithm for Boltzmann machines, Cogn. Sci. 9 (1) (1985) 147–169, http://dx.doi.org/10.1016/S0364-0213(85)80012-4.

[18] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780, http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[19] J.J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, Proc. Natl. Acad. Sci. 79 (8) (1982) 2554–2558, http://dx.doi.org/10.1073/pnas.79.8.2554, URL arXiv:https://www.pnas.org/content/79/8/2554.full.pdf.

[20] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507, http://dx.doi.org/10.1126/science.1127647.

[21] S. Li, W. Song, L. Fang, Y. Chen, P. Ghamisi, J.A. Benediktsson, Deep learning for hyperspectral image classification: An overview, IEEE Trans. Geosci. Remote Sens. 57 (9) (2019) 6690–6709, http://dx.doi.org/10.1109/TGRS.2019.2907932.

[22] V. Carletti, A. Greco, G. Percannella, M. Vento, Age from faces in the deep learning revolution, IEEE Trans. Pattern Anal. Mach. Intell. 42 (9) (2020) 2113–2132, http://dx.doi.org/10.1109/TPAMI.2019.2910522.

[23] T.T. Nguyen, N.D. Nguyen, S. Nahavandi, Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications, IEEE Trans. Cybern. 50 (9) (2020) 3826–3839, http://dx.doi.org/10.1109/TCYB.2020.2977374.

[24] L. Lei, Y. Tan, K. Zheng, S. Liu, K. Zhang, X. Shen, Deep reinforcement learning for autonomous Internet of Things: Model, applications and challenges, IEEE Commun. Surv. Tutor. 22 (3) (2020) 1722–1760, http://dx.doi.org/10.1109/COMST.2020.2988367.

[25] C. Athanasiadis, E. Hortal, S. Asteriadis, Audio–visual domain adaptation using conditional semi-supervised generative adversarial networks, Neurocomputing 397 (2020) 331–344, http://dx.doi.org/10.1016/j.neucom.2019.09.106.

[26] I. Rasheed, F. Hu, L. Zhang, Deep reinforcement learning approach for autonomous vehicle systems for maintaining security and safety using LSTM-GAN, Veh. Commun. 26 (2020) 100266, http://dx.doi.org/10.1016/j.vehcom.2020.100266.

[27] P. Wang, E. Fan, P. Wang, Comparative analysis of image classification algorithms based on traditional machine learning and deep learning, Pattern Recognit. Lett. (2020) http://dx.doi.org/10.1016/j.patrec.2020.07.042.

[28] M. Pikuliak, M. Šimko, M. Bieliková, Cross-lingual learning for text processing: A survey, Expert Syst. Appl. (2020) 113765, http://dx.doi.org/10.1016/j.eswa.2020.113765.

[29] W.G. Hatcher, W. Yu, A survey of deep learning: Platforms, applications and emerging research trends, IEEE Access 6 (2018) 24411–24432, http://dx.doi.org/10.1109/ACCESS.2018.2830661.

[30] X. Yuan, P. He, Q. Zhu, X. Li, Adversarial examples: Attacks and defenses for deep learning, IEEE Trans. Neural Netw. Learn. Syst. 30 (9) (2019) 2805–2824, http://dx.doi.org/10.1109/TNNLS.2018.2886017.

[31] M. Pawlicki, M. Choraś, R. Kozik, Defending network intrusion detection systems against adversarial evasion attacks, Future Gener. Comput. Syst. 110 (2020) 148–154, http://dx.doi.org/10.1016/j.future.2020.04.013, URL http://www.sciencedirect.com/science/article/pii/S0167739X20303368.

[32] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, X. Yi, A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability, Comp. Sci. Rev. 37 (2020) 100270, http://dx.doi.org/10.1016/j.cosrev.2020.100270, URL http://www.sciencedirect.com/science/article/pii/S1574013719302527.

[33] X. Zhang, Y. Zhou, S. Pei, J. Zhuge, J. Chen, Adversarial examples detection for XSS attacks based on generative adversarial networks, IEEE Access 8 (2020) 10989–10996, http://dx.doi.org/10.1109/ACCESS.2020.2965184.

[34] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Proceedings of the 27th International Conference on Neural Information Processing Systems, vol. 2, NIPS'14, MIT Press, Cambridge, MA, USA, 2014, pp. 2672–2680.

[35] H. Li, S. Zhou, W. Yuan, J. Li, H. Leung, Adversarial-example attacks toward Android malware detection system, IEEE Syst. J. 14 (1) (2020) 653–656, http://dx.doi.org/10.1109/JSYST.2019.2906120.

[36] K. Grosse, N. Papernot, P. Manoharan, M. Backes, P. McDaniel, Adversarial examples for malware detection, in: S.N. Foley, D. Gollmann, E. Snekkenes (Eds.), Computer Security – ESORICS 2017, Springer International Publishing, Cham, 2017, pp. 62–79, http://dx.doi.org/10.1007/978-3-319-66399-9_4.

[37] Q. Yan, F.R. Yu, Q. Gong, J. Li, Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges, IEEE Commun. Surv. Tutor. 18 (1) (2016) 602–622, http://dx.doi.org/10.1109/COMST.2015.2487361.

[38] S. Dong, K. Abbas, R. Jain, A survey on distributed denial of service (DDoS) attacks in SDN and cloud computing environments, IEEE Access 7 (2019) 80813–80828, http://dx.doi.org/10.1109/ACCESS.2019.2922196.

[39] N. Agrawal, S. Tapaswi, Defense mechanisms against DDoS attacks in a cloud computing environment: State-of-the-art and research challenges, IEEE Commun. Surv. Tutor. 21 (4) (2019) 3769–3795, http://dx.doi.org/10.1109/COMST.2019.2934468.

[40] Z. Li, H. Jin, D. Zou, B. Yuan, Exploring new opportunities to defeat low-rate DDoS attack in container-based cloud environment, IEEE Trans. Parallel Distrib. Syst. 31 (3) (2020) 695–706, http://dx.doi.org/10.1109/TPDS.2019.2942591.

[41] S. Velliangiri, H.M. Pandey, Fuzzy-Taylor-elephant herd optimization inspired deep belief network for DDoS attack detection and comparison with state-of-the-arts algorithms, Future Gener. Comput. Syst. 110 (2020) 80–90, http://dx.doi.org/10.1016/j.future.2020.03.049, URL http://www.sciencedirect.com/science/article/pii/S0167739X19332388.

[42] G.S. Kushwah, V. Ranga, Voting extreme learning machine based distributed denial of service attack detection in cloud computing, J. Inform. Secur. Appl. 53 (2020) 102532, http://dx.doi.org/10.1016/j.jisa.2020.102532, URL http://www.sciencedirect.com/science/article/pii/S2214212619304016.

[43] S. Akcay, A. Atapour-Abarghouei, T.P. Breckon, GANomaly: Semi-supervised anomaly detection via adversarial training, in: C.V. Jawahar, H. Li, G. Mori, K. Schindler (Eds.), Computer Vision – ACCV 2018, Springer International Publishing, Cham, 2019, pp. 622–637, http://dx.doi.org/10.1007/978-3-030-20893-6_39.
[44] N. Martins, J.M. Cruz, T. Cruz, P. Henriques Abreu, Adversarial machine learning applied to intrusion and malware scenarios: A systematic review, IEEE Access 8 (2020) 35403–35419, http://dx.doi.org/10.1109/ACCESS.2020.2974752.
[45] J. Singh, S. Behal, Detection and mitigation of DDoS attacks in SDN: A comprehensive review, research challenges and future directions, Comp. Sci. Rev. 37 (2020) 100279, http://dx.doi.org/10.1016/j.cosrev.2020.100279.
[46] J. C. Correa Chica, J.C. Imbachi, J. F. Botero Vega, Security in SDN: A comprehensive survey, J. Netw. Comput. Appl. 159 (2020) 102595, http://dx.doi.org/10.1016/j.jnca.2020.102595.
[47] P. Aitken, B. Claise, B. Trammell, Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information, Tech. Rep., (7011) RFC Editor, 2013, URL https://tools.ietf.org/html/rfc7011.
[48] M.L. Proença Jr., C. Coppelmans, M. Bottoli, A. Alberti, L.S. Mendes, The Hurst parameter for digital signature of network segment, in: Telecommunications and Networking - ICT 2004: 11th International Conference on Telecommunications, Fortaleza, Brazil, August 1-6, 2004. Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 772–781, http://dx.doi.org/10.1007/978-3-540-27824-5_103.
[49] L.F. Carvalho, T. Abrao, L. Mendes, M.L. Proença Jr., An ecosystem for anomaly detection and mitigation in software-defined networking, Expert Syst. Appl. 104 (2018) 121–133, http://dx.doi.org/10.1016/j.eswa.2018.03.027, URL http://www.sciencedirect.com/science/article/pii/S0957417418301726.
[50] S. Garg, K. Kaur, N. Kumar, G. Kaddoum, A.Y. Zomaya, R. Ranjan, A hybrid deep learning-based model for anomaly detection in cloud datacenter networks, IEEE Trans. Netw. Serv. Manag. 16 (3) (2019) 924–935, http://dx.doi.org/10.1109/TNSM.2019.2927886.
[51] Y. Zhou, G. Cheng, S. Jiang, M. Dai, Building an efficient intrusion detection system based on feature selection and ensemble classifier, Comput. Netw. 174 (2020) 107247, http://dx.doi.org/10.1016/j.comnet.2020.107247, URL http://www.sciencedirect.com/science/article/pii/S1389128619314203.
[52] E.H.M. Pena, S. Barbon, J.J.P.C. Rodrigues, M.L. Proença, Anomaly detection using digital signature of network segment with adaptive ARIMA model and paraconsistent logic, in: 2014 IEEE Symposium on Computers and Communications, ISCC, 2014, pp. 1–6, http://dx.doi.org/10.1109/ISCC.2014.6912503.
[53] C.E. Shannon, A mathematical theory of communication, Bell Syst. Tech. J. 27 (3) (1948) 379–423, http://dx.doi.org/10.1002/j.1538-7305.1948.tb01338.x.
[54] M. Wang, Y. Lu, J. Qin, A dynamic MLP-based DDoS attack detection method using feature selection and feedback, Comput. Secur. 88 (2020) 101645, http://dx.doi.org/10.1016/j.cose.2019.101645.
[55] P. Dixit, S. Silakari, Deep learning algorithms for cybersecurity applications: A technological and status review, Comp. Sci. Rev. 39 (2021) 100317, http://dx.doi.org/10.1016/j.cosrev.2020.100317, URL https://www.sciencedirect.com/science/article/pii/S1574013720304172.
[56] M.P. Novaes, L.F. Carvalho, J. Lloret, M.L. Proença, Long short-term memory and fuzzy logic for anomaly detection and mitigation in software-defined network environment, IEEE Access 8 (2020) 83765–83781, http://dx.doi.org/10.1109/ACCESS.2020.2992044.
[57] F. Chollet, et al., Keras, 2015, https://keras.io.
[58] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, Tensorflow: Large-scale machine learning on heterogeneous systems, 2015, Software available from tensorflow.org. URL http://tensorflow.org/.
[59] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (56) (2014) 1929–1958, URL http://jmlr.org/papers/v15/srivastava14a.html.
[60] Y. Wang, J. Liu, J. Mišić, V.B. Mišić, S. Lv, X. Chang, Assessing optimizer impact on DNN model sensitivity to adversarial examples, IEEE Access 7 (2019) 152766–152776, http://dx.doi.org/10.1109/ACCESS.2019.2948658.
[61] I. Sharafaldin, A.H. Lashkari, S. Hakak, A.A. Ghorbani, Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy, in: 2019 International Carnahan Conference on Security Technology, ICCST, 2019, pp. 1–8, http://dx.doi.org/10.1109/CCST.2019.8888419.
[62] M. Team, Mininet overview, 2020, Online Access: 2020-10-27 http://mininet.org/overview/. URL http://mininet.org/overview/.
[63] P. Floodlight, Floodlight, 2020, Online Access: 2020-10-27 http://www.projectfloodlight.org/. URL http://www.projectfloodlight.org/.
[64] P. Biondi, The Scapy community, 2020, Scapy Online Access: 2020-10-27 http://www.secdev.org/projects/scapy/. URL http://www.secdev.org/projects/scapy/.
[65] M.V. de Assis, L.F. Carvalho, J.J Rodrigues, J. Lloret, M.L. Proença, Near real-time security system applied to SDN environments in IoT networks using convolutional neural network, Comput. Electr. Eng. 86 (2020) 106738, http://dx.doi.org/10.1016/j.compeleceng.2020.106738.
[66] R. Priyadarshini, R.K. Barik, A deep learning based intelligent framework to mitigate DDoS attack in fog environment, J. King Saud Univ., Comput. Inf. Sci. (2019) http://dx.doi.org/10.1016/j.jksuci.2019.04.010.

Matheus P. Novaes received the master's degree in computer science from the State University of Londrina, Brazil, where he is currently pursuing the Ph.D. degree with the Electrical Engineering Department. He has been a member of the research group Computer Networks and Data Communication in the Computer Science Department with the State University of Londrina (UEL). His research focus is on management and security of computer networks.

Luiz F. Carvalho received the Ph.D. degree in Electrical Engineering and Telecommunications from State University of Campinas in 2018. He completed his master's degree in Computer Science at State University of Londrina in 2014. He has experience in Computer Science with emphasis in Computer Networks and is part of the research group Computer Networks and Data Communication. His main research interests are management and security of computer networks and Software-defined.

Jaime Lloret received his B.Sc.+M.Sc. in Physics in 1997, his B.Sc.+M.Sc. in electronic Engineering in 2003 and his Ph.D. in telecommunication engineering (Dr. Ing.) in 2006. He is a Cisco Certified Network Professional Instructor. He worked as a network designer and administrator in several enterprises. He is currently Associate Professor in the Polytechnic University of Valencia. He is the Chair of the Integrated Management Coastal Research Institute (IGIC) and he is the head of the "Active and collaborative techniques and use of technologic resources in the education (EITACURTE)" Innovation Group. He is currently the chair of the Working Group of the Standard IEEE 1907.1. Since 2016 he is the Spanish researcher with highest h-index in the TELECOMMUNICATIONS journal list according to Clarivate Analytics Ranking. He is an IEEE Senior, ACM Senior and IARIA Fellow.

Mario Lemes Proença Jr. is an Associate Professor and leader of the research group that studies computer networks in the Computer Science Department at State University of Londrina (UEL), Brazil. He received the Ph.D. degree in Electrical Engineering and Telecommunications from State University of Campinas (UNICAMP) in 2005. He received the title of M.Sc degree in Computer Science from the Informatics Institute of Federal University of Rio Grande do Sul (UFRGS), in 1998. He has authored or coauthored over 110 papers in refereed international journals and conferences, books chapters, and one software register patent. His research interests include Computer Network, Network Operations, Management and Security and IT Governance. He has supervised 14 M.Sc. and four Ph.D. students. He has been a master's supervisor at computer science in State University of Londrina and Ph.D. supervisor in Department of Electrical Engineering at UEL.