Universidade Estadual de Londrina

PEDRO VITOR PIASSA

UTILIZAÇÃO DE *DEEP LEARNING* PARA DETECÇÃO DE ANOMALIAS EM REDES

LONDRINA
2019

PEDRO VITOR PIASSA

UTILIZAÇÃO DE *DEEP LEARNING* PARA DETECÇÃO DE ANOMALIAS EM REDES

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

Orientador: Prof. Dr. Mario Lemes Proença Jr.

LONDRINA

2019

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

P581u Piassa, Pedro Vitor.
Utilização de *deep learning* para detecção de anomalias em redes / Pedro Vitor Piassa. - Londrina, 2019.
111 f. : il.

Orientador: Mario Lemes Proença Junior.
Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Universidade Estadual de Londrina, Centro de Ciências Exatas, Graduação em Ciência da Computação, 2019.
Inclui bibliografia.

1. Deep Learning - TCC. 2. Long Short-Term Memory - TCC. 3. Gated Recurrent Unit - TCC. 4. Detecção de Anomalias em Redes - TCC. I. Proença Junior, Mario Lemes. II. Universidade Estadual de Londrina. Centro de Ciências Exatas. Graduação em Ciência da Computação. III. Título.

CDU 519

PEDRO VITOR PIASSA

UTILIZAÇÃO DE *DEEP LEARNING* PARA DETECÇÃO DE ANOMALIAS EM REDES

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

BANCA EXAMINADORA

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

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

Prof. Msc. Fabio Cezar Martins
Universidade Estadual de Londrina

Londrina, 24 de novembro de 2019.

Dedico este trabalho a todas as pessoas que impactaram positivamente a minha vida durante a graduação. Principalmente a meu irmão Ulisses, que sempre me inspirou a seguir este caminho.

AGRADECIMENTOS

Meus agradecimentos vão primeiramente à minha família, por terem me proporcionado as condições necessárias para que eu pudesse me manter no curso; especialmente a minha mãe Solange, que sempre foi refúgio nos momentos de tristeza. Também ao meu irmão Ulisses, que foi fonte de inspiração para mim, desde que eu era pequeno.

Agradeço as amizades que fiz durante o curso, que contribuíram para a formação de quem sou hoje, por todos os momentos divertidos que passamos juntos, e também aos momentos de dificuldade que foram superados em equipe.

Meus sinceros agradecimentos a todos os professores que tive durante a graduação, principalmente aos professores Eliandro Rodrigues Cirilo e Ewerton da Silva Lemes pela imensa paciência que tiveram comigo em minhas constantes dúvidas durante os primeiros períodos do curso. Aos professores Adilson Luiz Bonifácio e Bruno Bogaz Zarpelão pela ótima didática e pelas aulas incríveis e de teor fascinante, que ajudaram a formar a maneira como penso durante processos de resolução de problemas. E ao professor Luiz Fernando Carvalho, por toda a ajuda que me forneceu durante a execução deste trabalho.

Por fim, agradeço a todas as pessoas que tiveram influência direta em minha vida e me ajudaram a cumprir meus objetivos, contribuindo para que eu fosse capaz de chegar onde cheguei; eu não teria conseguido sem a ajuda de todos vocês.

"It is the obvious which is so difficult to see most of the time. People say 'It's as plain as the nose on your face.' But how much of the nose on your face can you see, unless someone holds a mirror up to you?
(Isaac Asimov)

PIASSA, P. V.. **Utilização de *Deep Learning* para detecção de anomalias em redes**. 2019. 112f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2019.

RESUMO

Com a veloz evolução da internet e dos dispositivos interconectados, os métodos de ataque em ambientes de rede acabam tornando-se cada vez mais robustos. Sistemas de Detecção de Intrusão são empregados como mecanismo de defesa, porém, é necessário que estes possuam uma boa metodologia de identificação. Sabendo disso, este trabalho tem como objetivo pesquisar diferentes métodos de *Deep Learning* que possam ser aplicados na detecção de anomalias em redes para a criação de uma metodologia de defesa.

Palavras-chave: Deep Learning, Long Short-Term Memory, Gated Recurrent Unit, Detecção de Anomalias em Redes

PIASSA, P. V.. **Using Deep Learning for network anomaly detection**. 2019. 112p. Final Project (Bachelor of Science in Computer Science) – State University of Londrina, Londrina, 2019.

ABSTRACT

Due to the quick growth of internet and interconnected devices, web environment attacks has become more robust over time. Even though Intrusion Detection Systems are used as a defense mechanism, it is crucial that they have a strong identification approach. With that in mind, this work aims to research several Deep Learning methods that can be applied to network anomaly detection to build up a defense methodology.

Keywords: Deep Learning, Long Short-Term Memory, Gated Recurrent Unit, Network Intrusion Detection

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ARIMA	*Autoregressive Integrated Moving Average*
BLSTM	*Bi-Directional Long Short-Term Memory*
CNN	*Convolutional Neural Network*
DBN	*Deep Belief Network*
DDoS	*Distributed Denial of Service*
DL	*Deep Learning*
DoS	*Denial of Service*
DSAE	*Deep Stacked Auto-Encoder*
DSNSF	*Digital Signature of Network Segment Using Flow Analysis*
DWT	*Discrete Wavelet Transform*
FNN	*Feedforward Neural Network*
GRU	*Gated Recurrent Unit*
ICMP	*Internet Control Message Protocol*
IDS	*Intrusion Detection System*
IoT	*Internet of Things*
LSTM	*Long Short-Term Memory*
ML	*Machine Learning*
MLP	*Multilayer Perceptron*
NMSE	*Normal Mean Squared Error*
PCA	*Principal Component Analysis*
PNN	*Probabilistic Neural Network*
PSO	*Particle Swarm Optimization*
R2U	*Remote To User*
RF	*Random Forest*

RFE	*Recursive Feature Elimination*
RMSE	*Root Mean Squared Error*
RNN	*Recurrent Neural Network*
SDN	*Software Defined Network*
TSDL	*Two-Stage Deep Learning*
U2R	*User To Root*
UDP	*User Datagram Protocol*

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos é possível observar um aumento vertiginoso no uso da internet e no desenvolvimento de novas tecnologias que a empregam, permitindo com que as pessoas estejam cada vez mais interconectadas e mais dependentes da utilização da rede. Este crescimento é um dos principais fatores que contribui para um aumento do tráfego total gerado [1].

Entretanto, juntamente deste aumento do tráfego, começam a surgir alguns problemas relacionados à segurança na internet, que é importante objeto de estudo atualmente, já que mecanismos de ataque têm, por consequência do progresso, se tornado cada vez mais complexos [2][3][1][4]. Devido a isso, é necessário o uso de métodos mais modernos de detecção de anomalias em redes, capazes de passar alta confiabilidade e boa performance [3][4], que apresentem boas taxas de reconhecimento e baixa taxa de falsos positivos ao analisar grandes quantidades de dados.

Ainda é difícil definir de forma unânime o conceito de anomalia num ambiente de redes, fazendo com que o processo de definição de normalidade se torne parte do desenvolvimento de um mecanismo de detecção de anomalias [5]. Todavia, a definição mais comum para anomalias é dada a partir da observação de algum desvio no comportamento padrão de um fluxo de dados, de forma que as irregularidades encontradas podem ser relacionadas às falhas em dispositivos, invasões ao sistema ou ataques específicos, como negação de serviço ou escaneamento de portas.

Uma das principais anomalias que fazem parte do cenário atual de redes é o Ataque Distribuído de Negação de Serviço (*Distributed Denial of Service, DDoS*) [6], o qual apresenta grande ameaça à internet, considerando o momento evolutivo atual em que a mesma se encontra. O DDoS é caracterizado pela realização de um grande número de requisições, vindas de várias máquinas distintas, geralmente comprometidas [6], a um determinado serviço de rede, fazendo com que o mesmo se torne incapaz de aceitar requisições realizadas por usuários reais.

Os métodos mais comuns utilizados para proteger um ambiente de ataques externos, são os *firewalls*. Porém, estas soluções não são mais suficientes para proteger uma rede, fazendo com que seja importante a busca por métodos de defesa que sejam mais efetivos, quando na presença de uma situação anômala. Recentemente as pesquisas de segurança da informação estão mais voltadas para o desenvolvimento de métodos que utilizam técnicas computacionais mais modernas para detecção e mitigação de ataques.

A partir deste contexto, surge a necessidade de implementar mecanismos de detecção de anomalias que sejam capazes de se adaptar à complexidade das mesmas e à

grande quantidade de informações produzidas. Uma proposta da área de Segurança da Informação para lidar com anomalias voltadas para negação de serviço, entre outros tipos de anomalias em redes, é o Sistema de Detecção de Intrusão (*Intrusion Detection System, IDS*) [7][8], que é uma ferramenta cuja função é analisar o tráfego da rede em que é empregada e a partir disto tirar conclusões a respeito deste tráfego; os resultados indicam se a rede apresenta ou não uma anomalia, que neste caso é caracterizada por um desvio no comportamento padrão do tráfego.

Implementações de IDS são geralmente voltadas para identificação de padrões encontrados em um tráfego de redes [9]. Assim que um ataque é detectado, é enviado um sinal ao administrador da rede, informando a situação. O supervisor pode, por sua vez, decidir se irá mitigar o ataque ou interromper a comunicação.

Considerando o atual cenário das pesquisas relacionadas à segurança, este trabalho tem como objetivo estudar métodos de *Deep Learning* aplicados na detecção de anomalias em redes, partindo da definição de redes neurais e buscando apresentar os conceitos de redes neurais profundas, suas principais metodologias, e como o *Deep Learning* pode ser aplicado para solucionar os problemas atuais na esfera da segurança online.

Este trabalho propõe um estudo de métodos de *Deep Learning* aplicados na detecção de anomalias em redes e está organizado como segue: o Capítulo 2 faz uma breve descrição de como funciona um Sistema de Detecção de Intrusão e discorre sobre os principais conceitos de aprendizado de máquina e aprendizado profundo, e como estes conceitos são aplicados em detecção de anomalias em redes. Ainda neste capítulo, estão inclusas informações a respeito do modelo de aprendizado profundo que foi objeto principal de estudo deste trabalho, as Redes Neurais Recorrentes. O Capítulo 3 apresenta trabalhos correlatos produzidos nas áreas de análise de tráfego, aprendizado profundo e redes neurais recorrentes; além de outros métodos de *deep learning*. O Capítulo 4 mostra os princípios e metodologias para o desenvolvimento da pesquisa. O Capítulo 5 apresenta os resultados de um estudo de caso, obtidos a partir do treinamento da rede neural LSTM, de forma que estes resultados são discutidos no Capítulo 6. Em seguida, o Capítulo 7 mostra os resultados obtidos pela rede neural GRU e os compara com os resultados da LSTM. O Capítulo 8 faz uma análise dos resultados da GRU ainda comparando-os com os resultados da LSTM. E por fim, o Capítulo 9 apresenta as contribuições esperadas e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, serão apresentados os conceitos que englobam anomalias em rede, segurança e aprendizado de máquina, necessários para o desenvolvimento deste trabalho.

2.1 Ataques de Negação de Serviço

Ataques de negação de serviço (*Denial of Service*, DoS) são um dos tipos mais problemáticos de anomalias em redes, pois o objetivo deste tipo de ataque é fazer com que um determinado serviço na internet fique totalmente ou parcialmente congestionado, fazendo com que usuários reais não consigam acessá-lo [10], portanto, fica evidente que ataques desta ordem afetam diretamente uma das principais promessas de qualidade de serviço (*Quality of Service*, QoS), que é a disponibilidade [11]. Embora esta classe de anomalias seja capaz de causar estragos em grande escala, ainda fazem parte de um campo de pesquisa bastante desafiador, devido principalmente, à dificuldade na mitigação destes ataques [12].

O alto grau de periculosidade de ataques de negação de serviço é devido ao fato de que a maioria deles não necessita realizar a exploração de falhas específicas em software ou hardware, mas consistem no simples envio de quantidades muito grandes de pacotes de dados para um determinado serviço online, com o objetivo de torná-lo indisponível para usuários legítimos [6].

2.1.1 Ataque Distribuído de Negação de Serviço

Um ataque distribuído de negação de serviço (*Distributed Denial of Service*, DDoS), é uma versão mais poderosa de um Ataque de Negação de serviço provindo de uma única máquina ou dispositivo conectado na internet; neste caso, são utilizados diversos dispositivos em rede para a realização do ataque. Alguns tipos de ataques de negação de serviço são os ataques de *SYN flood*, *ICMP flood*, *UDP flood* e *Smurf*. Anomalias deste tipo visam inundar uma rede com uma quantidade muito grande de tráfego, tendo como objetivo prejudicar o funcionamento normal da mesma [13].

Um ataque distribuído, por sua vez, é considerado um dos tipos mais danosos na esfera das redes computacionais [14], pois consiste em não apenas uma máquina, mas diversas máquinas ou dispositivos disparando pacotes de dados contra um servidor, domínio, ou serviço na Internet, como mostra a figura 1. Geralmente, as máquinas adicionais que contribuem para o ataque, estão sob o controle de um invasor [13].

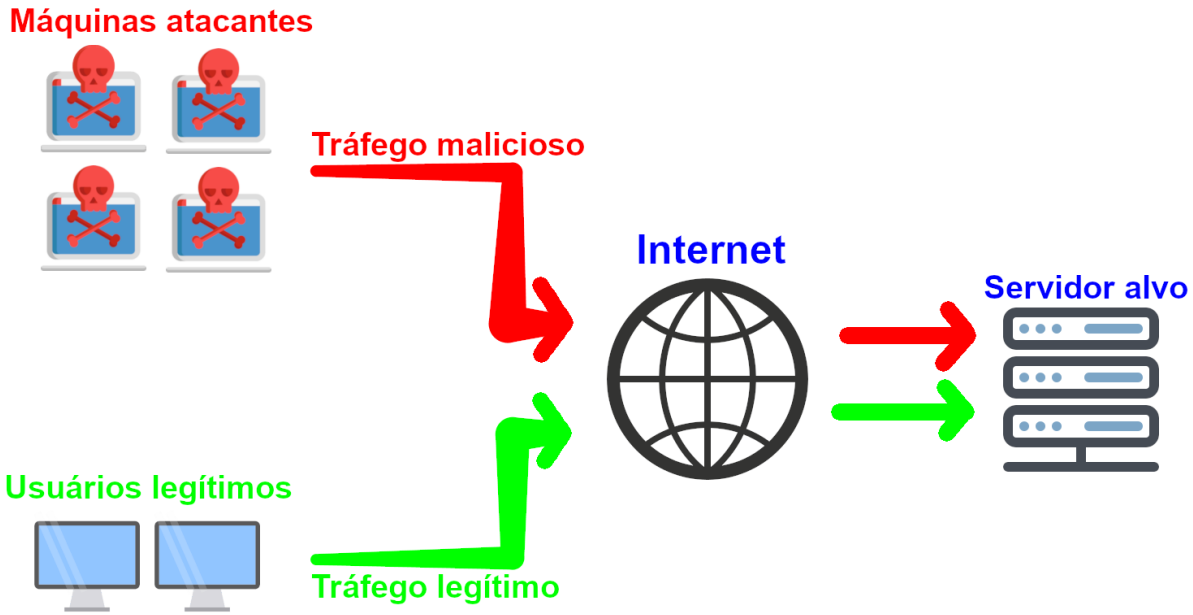


Figura 1 – Representação de um ataque DDoS. Fonte: próprio autor.

2.1.2 Botnet

O que dá a característica distribuída a um ataque de negação de serviço é o uso de diversos dispositivos que atacam um serviço online, de forma que estes dispositivos são, na maioria das vezes, comprometidos por um *malware*. A ferramenta maliciosa em questão permite o acesso remoto às máquinas infectadas que estejam conectadas na internet.

Uma *botnet* consiste em vários dispositivos infectados, um servidor de comando e controle (*Command and Control*, C&C), e um *botmaster*, que é quem tem acesso ao servidor de controle das *botnets* (figura 2) [15]. Na maioria das vezes, as vítimas acometidas não se dão conta de que suas máquinas estão comprometidas e que fazem parte de uma rede de *bots* [6].

2.1.2.1 Internet das Coisas

O conceito de Internet das Coisas (*Internet of Things*, IoT) é bastante recente, e é definido a partir do uso de dispositivos físicos em diversas áreas da tecnologia, como automação residencial e monitoramento, de forma que tais objetos estão conectados em rede e são capazes de interagir uns com os outros. A aplicação deste novo paradigma em diferentes campos é capaz de trazer diversos benefícios, entretanto, também carregam consigo novos desafios relacionados à segurança em redes [16].

A falta de segurança em dispositivos IoT é um chamariz para hackers que têm como objetivo encontrar e abusar das falhas de segurança existentes nestes aparelhos. A ameaça de falhas neste tipo de ferramenta se dá principalmente por possuírem a capacidade de serem amplamente distribuídas, podendo ser transformadas em *bots*, de forma a contribuir

para a sintetização das *botnets* [17][18].

Um software malicioso chamado *Mirai*, cujo código fonte foi publicado num fórum *hacker*, tem como foco invadir dispositivos IoT que possuem falhas de segurança [19]. Um dos casos mais notórios ocorreu em 2016, quando um provedor de DNS chamado *Dyn* registrou em sua rede tráfegos de até 1,2 Tbps, tal que o resultado deste ataque foi a indisponibilidade de serviços como o *Reddit*, *Github*, *Twitter* e *Netflix* [18][19].

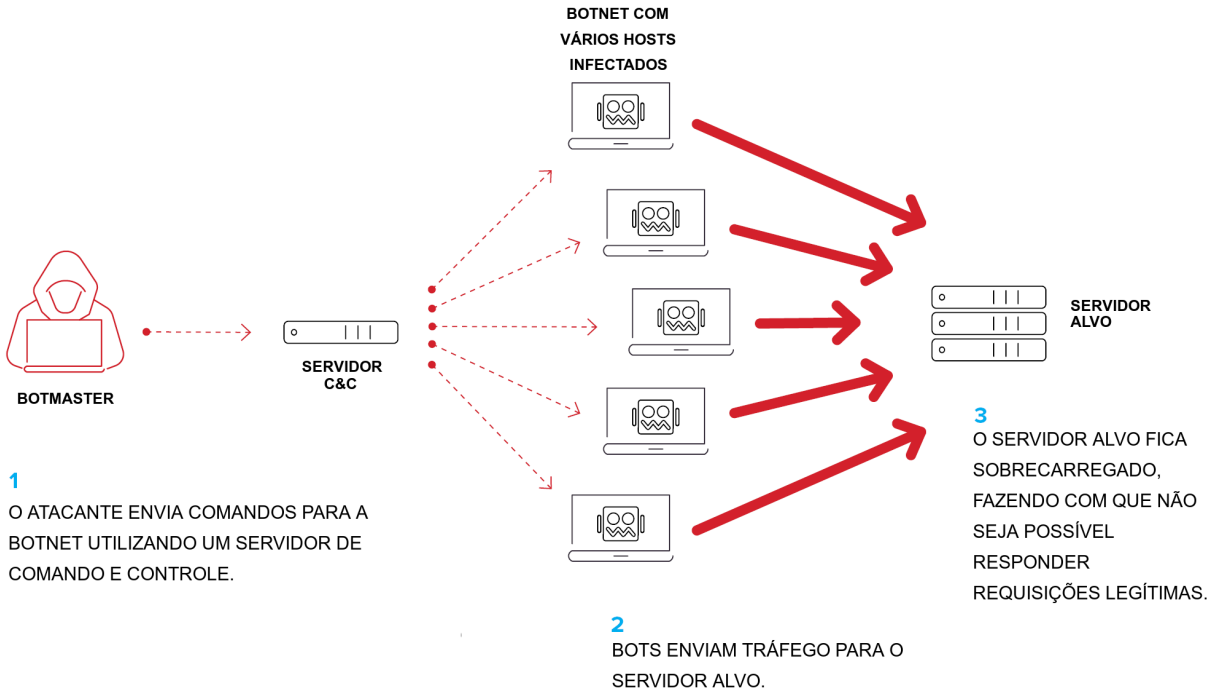


Figura 2 – Ataque DDoS realizado com o auxílio de uma *botnet*. Fonte: adaptado de [20]

2.2 Redes Definidas por Software

Redes Definidas por Software (*Software Defined Networks*, SDN) é um paradigma emergente cujo objetivo é simplificar a criação e a introdução de novas abstrações no âmbito de redes, facilitando sua gerência e evolução [21]. E é devido ao acelerado crescimento do número de dispositivos interconectados, o qual contribui para ambientes de rede cada vez mais complexos, que surge a necessidade de diferentes abordagens para o tratamento deste problema; Redes Definidas por Software é uma destas abordagens [22][23]. Entretanto, devido ao fato das redes SDN ainda estarem em desenvolvimento, são necessários diversos testes, por ser uma metodologia que trará mudanças significativas no processo de sintetização de uma arquitetura de redes [24].

2.2.1 Objetivos

Este paradigma tem como característica isolar o plano de controle, que é onde são realizadas as decisões a respeito do gerenciamento do tráfego, do plano de dados,

que é onde acontece o redirecionamento de pacotes, oferecendo uma unidade centralizada que é utilizada para realizar o controle da rede [24]. Devido a isso, o método oferece uma boa simplificação da arquitetura de uma rede, e torna mais simples a formulação de métodos de segurança para redes deste tipo, principalmente pelo fato de que o tráfego fica centralizado no controlador. A centralização do tráfego neste tipo de paradigma faz com que seja mais simples armar um plano de defesa, pois, pela mesma razão do controlador ser um único ponto de falha, é também o principal local a ser protegido, visto que fornece a visão central da rede [14]. Em [25] é proposto um sistema proativo voltado para ambientes SDN, capaz de identificar e mitigar anomalias em tempo real.

As vantagens da centralização dos dados, oferecida por este paradigma, envolvem, principalmente, a facilidade que um gerente de redes tem para implementar novas funcionalidades na rede, tais como roteamento; *switching*; sintetização de um *firewall*; tradução de endereços de rede; balanceamento de carga; entre outros [26].

2.2.2 O protocolo OpenFlow

O protocolo *OpenFlow*, promovido pela *Open Network Foundation*, tem como papel prover uma interface de comunicação entre a camada de controle, ou seja, o controlador SDN, e a infraestrutura, ou o *switch* SDN. Embora existam outros protocolos capazes de realizar este interfaceamento, o protocolo *OpenFlow* é o padrão dominante [26].

Sendo assim, o *OpenFlow* é um dos protocolos que implementa o conceito de SDN, provendo fluxos programáveis, de forma que o administrador da rede consegue definir o caminho que um pacote deverá percorrer, independente da topologia implementada [24], além de uma interface de comunicação entre o controlador SDN e o *switch* SDN. Além disso, este protocolo realiza a abstração os elementos da rede através da realização de tarefas como o controle de operações de encaminhamento, notificações de eventos e relatórios estatísticos [23].

2.3 Detecção de Anomalias

Anomalias, referidas como desvios no comportamento padrão da rede, podem ser geradas por diversas atividades irregulares, porém o aspecto mais importante para detecção de anomalias é a essência da mesma [27]. Uma anomalia pode ser caracterizada como sendo uma **anomalia pontual**, que indica um desvio no comportamento de um conjunto particular de dados; **anomalia contextual**, quando dados se comportam de forma anômala dentro de um determinado contexto; e **anomalia coletiva**, quando instâncias similares de dados possuem comportamento anormal dentro de todo o conjunto de dados da rede [27].

A importância de processos de detecção e mitigação de anomalias em rede se dá

com o objetivo de que aplicações online assegurem confidencialidade, integridade e disponibilidade de seus serviços, sendo estes os principais métodos garantidores de seguridade, fazendo parte dos pilares da segurança da informação, cujas definições são dadas a seguir [28].

- **Confidencialidade**: A confidencialidade de um serviço está relacionada com a privacidade dos dados que são armazenados no mesmo, de forma a garantir seu acesso apenas a usuários legítimos;
- **Integridade**: Integridade visa manter a consistência da aplicação de forma a garantir que quaisquer dados que circulem pela rede mantenham sempre as mesmas condições de quando foram gerados, sem que haja interferência externa que os modifiquem.
- **Disponibilidade**: Está atrelada à seguridade do acesso a quaisquer tipos de informações a qualquer momento, desde que a requisição venha de um usuário legítimo.

Diversos ataques podem ocorrer num ambiente de rede. Alguns dos principais tipos são os ataques de reconhecimento, também conhecidos como *Probing*; Usuário para super usuário (*User to Root, U2R*); Remoto para usuário (*Remote to User, R2U*); e os ataques de negação de serviço [27].

Ataques de reconhecimento não são diretamente prejudiciais, de forma que ocorrem antes de algum tipo de dano ser infligido contra um serviço. Os objetivos são simples: Coletar o máximo possível de informações, de forma a encontrar os tipos e quantidades de máquinas que estão conectadas numa rede, quais portas estão abertas nos dispositivos e quais serviços possuem conexões com essas portas. Os resultados deste tipo de ataque fornecem a um usuário malicioso informações valiosas a respeito do perfil da rede alvo, de forma que o invasor pode, a partir disso, definir quais ataques serão mais eficazes.

A classe de anomalias que englobam os ataques de negação de serviço é uma das mais perigosas [12]. Anomalias deste tipo têm como objetivo exaurir os recursos de um serviço alvo a partir do envio de grandes quantidades de requisições válidas para o mesmo.

Ataques *U2R* ocorrem quando um usuário já possui acesso a um sistema como usuário comum e busca, a partir disso, elevar seu nível de privilégio para obter acesso a ferramentas administrativas. A classe de ataques *R2U* se assemelha bastante com a *U2R*, entretanto, neste caso, o atacante obtém acesso remoto a um serviço partindo da exploração de vulnerabilidades presentes no mesmo, sem possuir legitimamente uma conta de usuário. Ataques deste tipo geralmente envolvem engenharia social ou força bruta para o descobrimento de senhas [27].

Alguns dos trabalhos devolvidos pelo grupo de pesquisa de redes de computadores do departamento de Computação da UEL que tem o objetivo de pesquisar sobre gerência,

segurança e detecção de intrusões e anomalias em redes de computadores, são os artigos [29] [30] [31] [32].

2.3.1 Sistema de Detecção de Intrusão

Um Sistema de Detecção de Intrusão (*Intrusion Detection System, IDS*) são mecanismos de defesa, vinculados a redes ou computadores, com o objetivo de realizar o processo de identificação de violações e intrusões a partir da análise do tráfego de entrada. IDS são aplicados como uma segunda linha de defesa de sistemas de rede, o *Firewall* ficando em primeiro (figura 3); entretanto, atacantes estão continuamente buscando falhas nestes sistemas, fazendo com que seja crucial a implementação de um bom IDS.

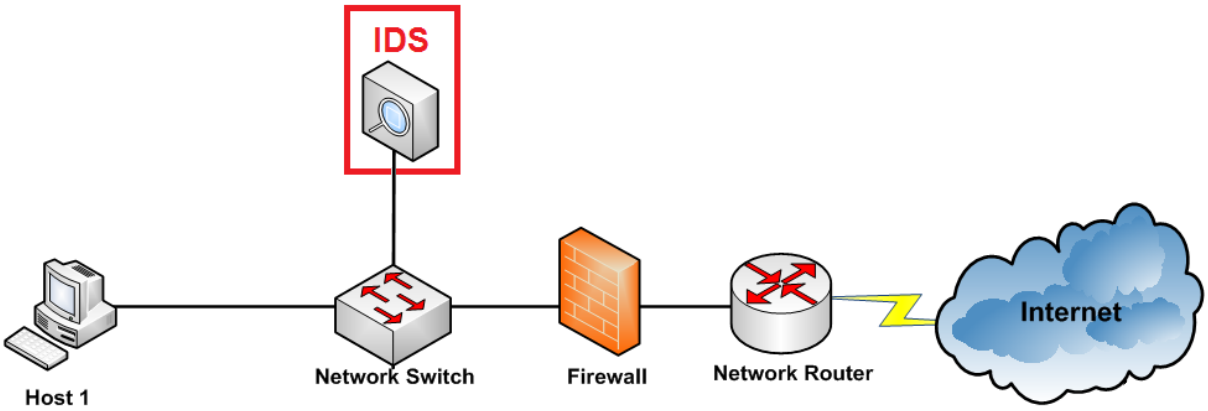


Figura 3 – Arquitetura de rede contendo um *firewall* e um IDS. Fonte: adaptado de [33]

No geral, Sistemas de Detecção de Intrusão contêm três principais componentes, sendo o primeiro deles um mecanismo para coleta dos fluxos de rede, tal que estes fluxos são definidos como sendo uma sequência de vários pacotes de IP [34]. Após esta coleta é realizada a seleção das características e finalmente um procedimento de classificação deve ser executado para decidir se o tráfego é considerado normal ou anômalo. A parte da detecção de anomalias do IDS pode ser feita baseada em assinaturas de tráfego de ataques já conhecidos, o que tende a ser bastante eficiente, porém está sujeito a falhar diante de tráfego anômalo desconhecido [7]. E é diante deste cenário que surge a necessidade de implementar um sistema de detecção de anomalias que opere de forma dinâmica e que seja capaz de aprender a identificar trafego anômalo e trafego não anômalo por meio de aprendizado de máquina.

Alguns exemplos de Sistemas de Detecção de Intrusão implementados que fazem uso de técnicas de aprendizado de máquina, são os apresentados em [35][36][37] e [38], em que o primeiro faz o uso de Teoria dos Jogos e Algoritmo Genético com Lógica Difusa para detecção e mitigação de ataques DDoS em redes SDN; o segundo trabalho também tem como foco a detecção de anomalias do tipo DDoS em ambiente SDN e utilizam Algoritmo Genético com Lógica Difusa. A metodologia de detecção proposta pelo terceiro artigo é

a de Transformada Discreta de *Wavelets* (*Discrete Wavelet Transform, DWT*) bem como um algoritmo baseado em Floresta Aleatória para identificação e mitigação de anomalias em âmbito de Redes Definidas por Software. No quarto trabalho, é proposta a criação de uma ferramenta que faz uso de lógica paraconsistente para detecção de anomalias e utiliza de Assinaturas Digitais de Segmento de rede Utilizando Análise de Fluxo (*DSNSF*) como perfil normal de tráfego.

2.3.2 Entropia de Shannon

O conceito de entropia é um termo muito comum na física, e é amplamente utilizado para discutir o grau de desordem de um sistema. Na estatística, o grau de entropia de um conjunto de elementos pode ser intuído a partir da probabilidade de ocorrência de um determinado evento naquele conjunto [39][40][41]. Outra maneira comum de definir entropia é a partir da quantidade de perguntas binárias (sim/não) necessárias, em média, para identificar um elemento qualquer, contido num conjunto de tamanho arbitrário [39].

A definição matemática da entropia de Shannon é dada pela equação a seguir [40]:

$$H_s = -\sum_{i=1}^{N} p_i log_2(p_i), \tag{2.1}$$

Na qual p_i é a probabilidade de ocorrer i e N é o número total de ocorrências no espaço amostral sendo analisado. O resultado de H_s varia entre 0 e $log_2(N)$, sendo que, o menor valor representa concentração máxima na distribuição, indicando que a ocorrência de i se dá para todos os casos. O maior valor representa dispersão máxima, de forma que a probabilidade de ocorrência de i é a mesma para qualquer i contido no espaço amostral.

2.3.2.1 Relação com anomalias em rede

A relação da entropia de Shannon com detecção de anomalias se dá de maneira bastante aparente quando se trata de ataques que concentram vários elementos num único alvo, exatamente como operam ataques do tipo DDoS. Levando em conta a natureza do ataque, em que diversas máquinas provindas de diferentes origens disparam contra um único alvo, ao observar o fluxo da rede durante o evento, nota-se a presença de vários endereços de IP e valores de portas de origem, disparando dados contra um único ponto da rede.

Considerando o cenário descrito acima, pode-se considerar a quantidade de IP e portas de destino como dois conjuntos distintos, tal que a existência de um IP neste conjunto indica que existe um outro, de origem, associado a ele; o raciocínio é análogo para o conjunto de portas de destino. Isto implica na existência de uma probabilidade p_i associada à ocorrência de cada elemento em cada um dos conjuntos.

Sabendo-se que existe uma grande quantidade de valores repetidos de IP e portas de destino (já que diferentes IP e portas de origem disparam contra um mesmo IP e porta de destino) ao realizar o cálculo da entropia de Shannon, os resultados obtidos para cada um dos conjuntos irão tender a 0, representando uma alta taxa de concentração de tráfego ocorrendo na rede [25][42]. Este fenômeno pode ser observado nas figuras 4 e 5, que representam, respectivamente, a entropia de IP e portas de destino em função do tempo, calculadas para um tráfego contendo um ataque DDoS entre 9:15 e 10:30.

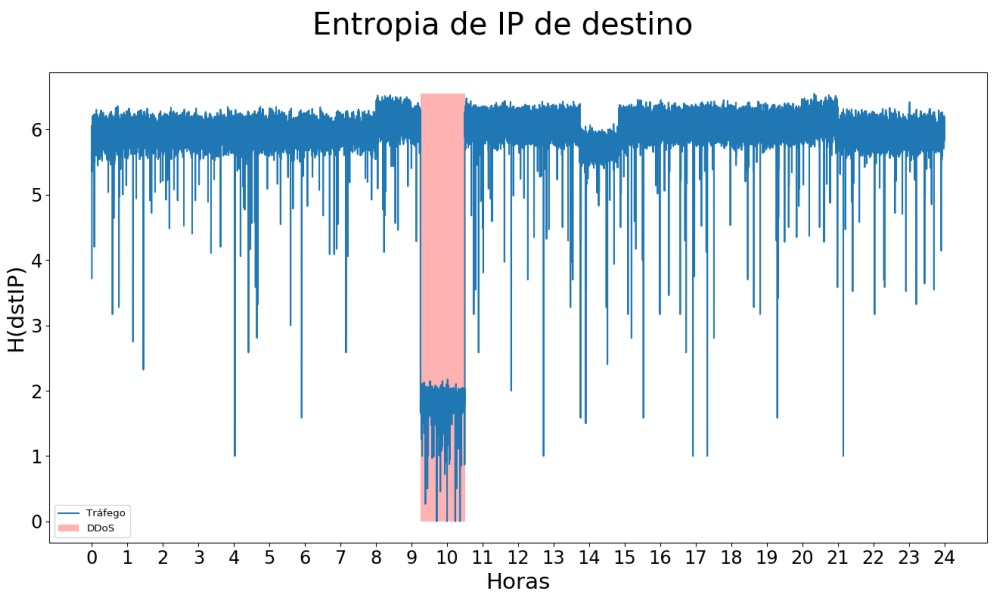


Figura 4 – Representação gráfica da entropia de Shannon calculada para a dimensão de IP de destino, ao longo de 24 horas. Fonte: próprio autor.

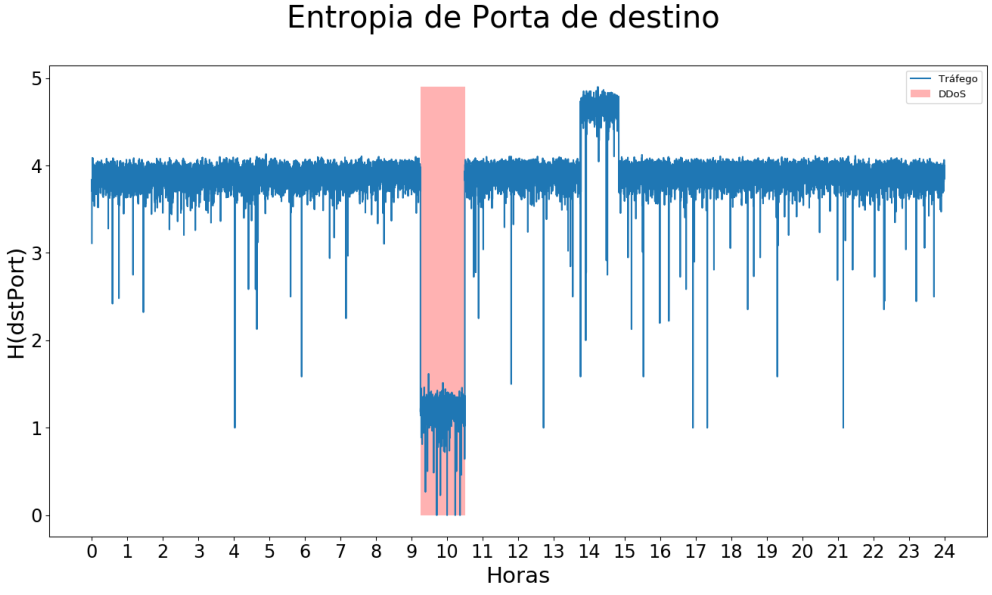


Figura 5 – Representação gráfica da entropia de Shannon calculada para a dimensão de portas de destino, ao longo de 24 horas. Fonte: próprio autor.

2.3.3 *Deep Learning* e detecção de anomalias em redes

Embora os Sistemas de Detecção de Intrusão tenham conquistado um papel importante na proteção de redes de computadores, existem preocupações com relação às metodologias atuais aplicadas para tal, principalmente devido à demanda de redes modernas. Tais preocupações envolvem principalmente o crescimento do volume de dados sendo transmitidos em ambientes *web*, cuja tendência é continuar aumentando [2]. Devido a isso, novas pesquisas na área de detecção de anomalia em redes estão sendo feitas e *Deep Learning* constitui uma abordagem promissora para auxiliar na implementação de sistemas de segurança [3].

Um dos desafios de métodos tradicionais é a detecção de trafego desconhecido, visto que abrangem 17% de todo o tráfego de redes. Métodos de análise que fazem uso de *Deep Learning* são capazes de identificar mais da metade deste tipo de tráfego [3].

2.4 Aprendizado de Máquina

Aprendizado de máquina (*Machine learning*) pode ser definido como um conjunto métodos estatísticos computacionais que consistem no uso de uma base de dados que descrevem determinado fenômeno. Estes algoritmos utilizam de experiência e treinamento para melhorar sua performance e realizar previsões bem definidas, com base nas informações disponíveis, para a resolução de problemas práticos [43][44][21].

Existem vários tipos de metodologias de aprendizado de máquina, uma delas é o de aprendizado supervisionado que refere-se a quando um algoritmo é alimentado com uma base de dados de treinamento contendo amostras rotuladas, podendo, assim, realizar previsões em pontos desconhecidos. Outro método é o de aprendizado não supervisionado, em que o algoritmo é alimentado exclusivamente com dados não rotulados e realiza previsões para pontos desconhecidos[43][44].

2.4.1 Tarefas de aprendizado

Algoritmos de aprendizado de máquina são especializados na resolução dos mais diversos tipos de problemas, sejam eles de caráter supervisionado ou não. As duas especializações mais comuns dos algoritmos de aprendizado de máquina são as de solucionar problemas do tipo classificatório ou regressivo [43].

2.4.1.1 Classificação

Problemas classificatórios são identificados pelos resultados de teor categórico, que são esperados do modelo. Estes são o tipo de problema em que os dados de saída possuem categorias bem definidas, de forma a produzirem resultados discretos. Um exemplo deste

tipo de problema é o de classificação de imagens com relação aos objetos contidos na mesma.

O tipo mais simples de classificação é a classificação linear, que consiste em separar um conjunto de dados em duas classes. A ideia matemática do modelo de classificação linear é o de se obter uma solução para uma equação de reta ($f(x) = ax + b$) a partir do cálculo do gradiente da mesma, com relação a a e b, como mostra a equação 2.2.

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial a} \\ \frac{\partial f(x)}{\partial b} \end{bmatrix} \tag{2.2}$$

A generalização do método para uma função $f(x)$ multivariada, cujos parâmetros são denotados pelo conjunto $\Theta = \{\theta_1, \theta_2, \cdots \theta_N\}$, de tamanho N, é exibida na equação 2.3;

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial \theta_1} \\ \frac{\partial f}{\partial \theta_2} \\ \frac{\partial f}{\partial \theta_3} \\ \vdots \\ \frac{\partial f}{\partial \theta_N} \end{bmatrix} \tag{2.3}$$

2.4.1.2 Regressão

Em regressão, as soluções desejadas são de modo que seja possível aproximar a relação entre as informações de entrada a partir de uma reta, de forma a produzir valores reais para cada item. Exemplos de problemas que requerem este tipo de abordagem são os de gênero preditivo em que, dadas as informações de entrada (ex.: vários dias contendo o comportamento do tráfego de uma rede até a hora atual), o modelo irá realizar uma previsão como saída (como o tráfego irá se comportar nas próximas horas).

A base dos problemas de regressão são oriundas da Estatística, na qual a regressão linear é uma maneira de aproximar, a partir de uma reta, o valor de uma variável y com base nos valores de um conjunto de variáveis x. Em *machine learning*, o problema é geralmente tratado como sendo um problema de aprendizado supervisionado, ou seja, os valores de y já são fornecidos e a ideia é, dado um conjunto de pontos (x, y), encontrar os parâmetros a e b da função da reta $f(x) = ax + b$ que melhor aproximarem os valores de y.

Cada um dos pontos do conjunto (x, y) podem ser utilizados para criar um sistema de equações lineares (Equação 2.4), o qual pode ser denotado na forma matricial, como

visto em 2.5, de maneira que formam a Equação 2.6.

$$\begin{aligned} ax_1 + b &= y_1 \\ ax_2 + b &= y_2 \\ ax_3 + b &= y_3 \\ &\vdots \\ ax_n + b &= y_n \end{aligned} \tag{2.4}$$

$$A = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \vec{m} = \begin{bmatrix} a \\ b \end{bmatrix} \vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} \tag{2.5}$$

$$A\vec{m} = \vec{y} \tag{2.6}$$

Assim, a partir da Equação 2.6, é possível derivar uma fórmula que calcula os parâmetros a e b a partir de cada um dos pontos do conjunto (x, y), como demonstrado a seguir:

Demonstração.

$$\begin{aligned} A\vec{m} &= \vec{y} \\ A^T A\vec{m} &= A^T\vec{y} \\ (A^T A)^{-1}(A^T A)\vec{m} &= (A^T A)^{-1}A^T\vec{y} \\ I\vec{m} &= (A^T A)^{-1}A^T\vec{y} \\ \vec{m} &= (A^T A)^{-1}A^T\vec{y} \end{aligned}$$

□

2.5 Redes Neurais

Modelos de redes neurais são inspirados na estrutura do cérebro humano, que pode ser interpretada como sendo um poderoso emaranhado de neurônios, cuja potência se dá pela grande quantidade de sinapses formadas entre si [45]. Um dos modelos mais simples de redes neurais que pode ser utilizado para descrever este paradigma, é o perceptron.

2.5.1 Perceptron

O perceptron é uma unidade básica de processamento, cuja função é servir de classificador linear binário, sendo capaz de separar um espaço de características em duas

classes distintas, utilizando uma linha reta (Figura 7). Seus valores de entrada podem ser primários ou podem vir de outros perceptrons anteriores, que são associados a este por meio de uma sinapse e um peso sináptico [45]. A figura 6 ilustra um perceptron cujos valores de entrada são denotados por $x_1, \cdots, x_m$; os pesos por $w_1, \cdots, w_m$ e a camada de saída é representada por $\hat{y}$.

Pesos sinápticos são valores que são multiplicados aos valores de saída de um neurônio que está passando informação adiante, de forma que exercem o papel de inibidores ou reforçadores de uma conexão sináptica entre o neurônio que envia um sinal e o neurônio que o recebe. Após a soma dos valores de entrada multiplicados pelos seus respectivos pesos sinápticos, o resultado é somado e passa por uma função de ativação, que produz um valor de classificação [45].

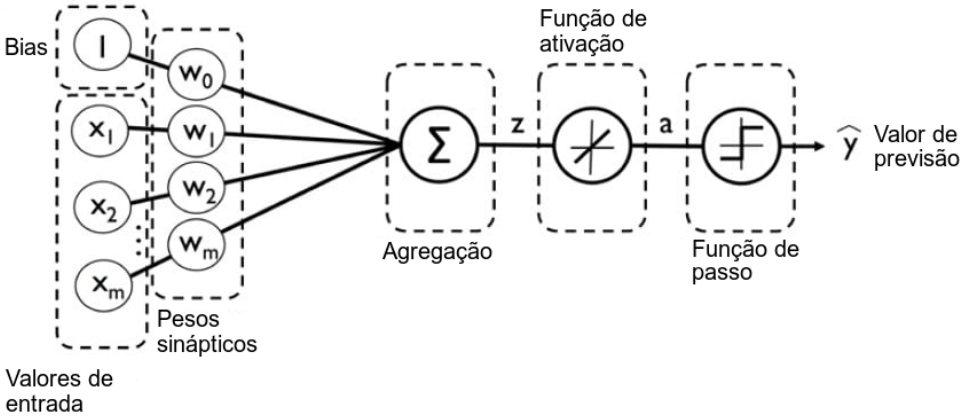


Figura 6 – Rede neural de um único perceptron, contendo m valores de entrada. Fonte: adaptado de [46]

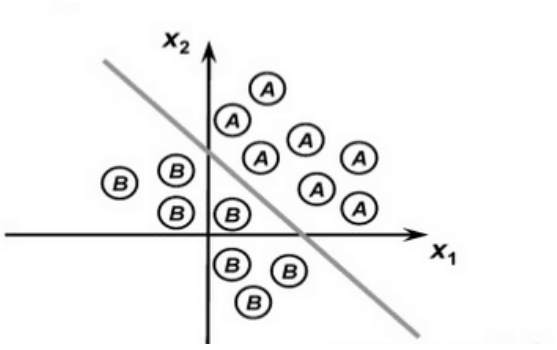


Figura 7 – Conjunto de dados linearmente separáveis. Fonte: [47]

O peso sináptico tem um papel importante no aprendizado da rede neural, pois é o valor dele quem será atualizado por meio da derivada da função de custo com relação ao peso do nó (método conhecido como *backpropagation*), multiplicada pela taxa de aprendizado, como mostra a Equação 2.7, na qual α é a taxa de aprendizado, C é a função de

custo e $w_{j,k}^{(l)}$ é um peso sináptico que conecta o nó k de uma camada l ao nó j da camada $l+1$ [48].

$$w_{j,k}^{(l)} = w_{j,k}^{(l)} - \frac{\partial C}{\partial w_{i,j}^{(l)}}\alpha \tag{2.7}$$

2.5.2 Perceptron Multicamadas

O conceito de perceptron de multicamadas é a base de uma rede neural mais robusta, capaz de estimar valores para resolver problemas que exigem soluções não lineares. Neste caso, a rede neural passa a ter múltiplos perceptrons acionados por sinapses, de forma que as camadas centrais, distribuídas entre a primeira e a última, são chamadas de camadas ocultas, ou *hidden layers*. O conceito será discutido novamente na seção 2.7.1.

A figura 8 mostra um perceptron multicamadas com n valores de entrada, uma camada oculta e m valores de saída.

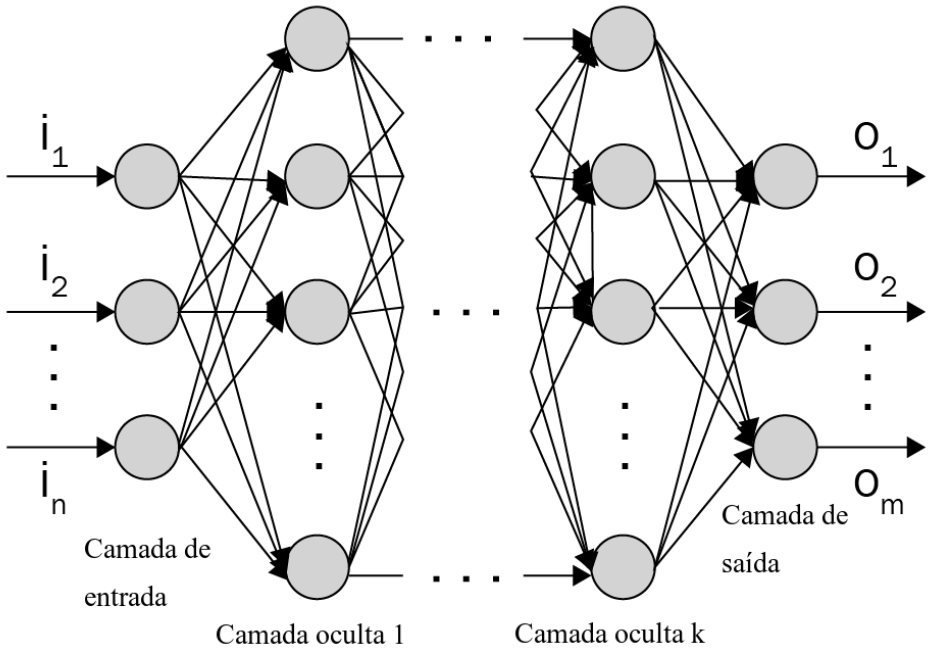


Figura 8 – Perceptron multicamadas. Fonte: adaptado de [49]

2.5.3 Funções de ativação

Funções de ativação são muito importantes em redes neurais, pois servem para limitar a amplitude do valor de saída de um neurônio, ou para definir se a informação que um neurônio está recebendo é relevante, podendo ativá-lo ou não [50]. Alguns exemplos de funções de ativação, que serão utilizadas na construção dos métodos RNN, LSTM e GRU, são a função *sigmoid*, comumente denotada por $\sigma(x)$, a tangente hiperbólica, denotada por $\tanh(x)$ e a função *softmax*.

2.5.3.1 Sigmoid

A função *sigmoid*, denotada pela Equação 2.8, produz como saída um valor pertencente ao intervalo $(0, 1)$, representando uma distribuição de probabilidade. É vantajosa quando aplicada em redes neurais treinadas com algoritmos de *backpropagation*, por ser facilmente distinguível [50].

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.8}$$

2.5.3.2 Tanh

Esta função é similar à *sigmoid* e pode ser definida pela razão entre o seno e o cosseno hiperbólicos, ou pela diferença e soma de exponenciais, como mostra a equação 2.9. Uma das principais diferenças entre a tangente hiperbólica e a função *sigmoid*, é o intervalo de sua imagem, que pertence ao intervalo $(-1, 1)$ [50]. Além disso, a tangente hiperbólica é normalmente utilizada nas camadas ocultas de uma rede neural, enquanto que a *sigmoid* é mais comumente utilizada em problemas de classificação binária, sendo localizada na camada de saída.

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{2.9}$$

Na figura 9 é possível observar o gráfico comparativo entre ambas as funções.

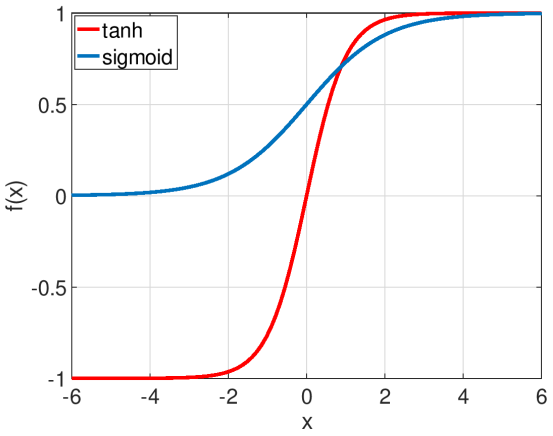


Figura 9 – Comparação entre as funções de ativação *sigmoid* e *tanh*. Fonte: próprio autor.

2.5.3.3 Softmax

A função *softmax* atua como uma função de ativação comumente usada em problemas de classificação e é aplicada na camada *logit*, isto é, a última camada do modelo, para calcular a probabilidade de uma classe num problema multivariado. O resultado é

um vetor contendo distribuição de probabilidade para cada elemento da saída, cuja soma é 1 [51][52]. Sua fórmula é denotada pela equação 2.10 [53].

$$softmax(y_i) = \frac{\exp(y_i)}{\sum_{j=1}^{n} \exp(y_j)}, \tag{2.10}$$

em que y é o vetor de saída.

2.6 Aprendizado Profundo

Deep Learning é uma subclasse da inteligência artificial que está em constante ascensão e pode ser aplicada em vários tipos de tecnologias que requerem grande volume de dados. Alguns exemplos são: reconhecimento de fala; reconhecimento de caligrafia; classificação de imagens, entre outras aplicações [54]. Uma arquitetura de *Deep Learning* é caracterizada por utilizar uma grande camada interna de módulos mais simples chamados de *hidden layer* em que todos, ou grande parte destes módulos, estão sujeitos a aprendizado, e no final desta sucessão produzem dados de saída [53]. Cada um dos módulos desta cadeia transforma seus dados de entrada (*inputs*) para aumentar tanto a seletividade quanto a invariância da representação. Com múltiplas camadas, um sistema é capaz de implementar funções extremamente complexas que sejam sensíveis a mínimos detalhes e insensitivos a grandes variações irrelevantes nos dados de entrada [55]. A Figura 10 contém a representação de uma rede neural de *Deep Learning*. A coluna mais à esquerda representa os dados de entrada; as colunas centrais simbolizam as várias camadas de abstração do método, ou *hidden layers*; e, por fim, o quadrado à direita descreve os dados de saída.

O progresso da técnica é devido, principalmente, à robustez que a mesma possui em reconhecimento de padrões complexos [56]. A tendência é que a área continue evoluindo, já que *Deep Learning* é um mecanismo que tira vantagem do crescente número de dados computacionais disponíveis, por efeito de ser uma metodologia que necessita de pouco trabalho manual para ser implementado, ou seja, requer uma menor interferência do ser humano no processo, diferentemente de métodos rasos de aprendizado de máquina e reconhecimento de padrões que demandam um cuidadoso processo de engenharia. Estas são uma das principais vantagens do *Deep Learning* em contraste com técnicas tradicionais de *Machine Learning* [55][56].

2.7 Métodos de *Deep Learning*

A seguir, serão apresentados os métodos mais comuns aplicados em redes neurais profundas.

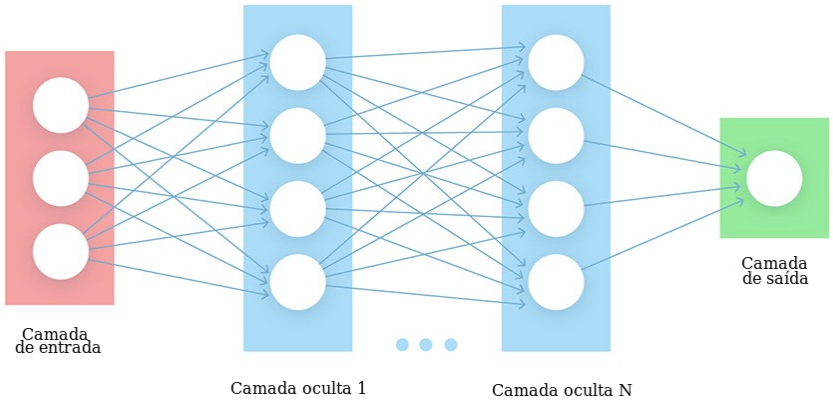


Figura 10 – Representação gráfica de uma Rede Neural Profunda. Fonte: adaptado de [57]

2.7.1 *Deep Feedforward Network*

Este método, também conhecido como *Feedforward Neural Network* ou Perceptron Multicamadas (*Multilayer Perceptron, MLP*) é o esquema mais comum e mais primordial das redes neurais profundas. O objetivo de uma rede neural *feedforward* é o de aproximar o valor de uma função f^*, podendo ser um classificador do tipo $y = f^*(x)$, para o qual, dado um valor x de entrada, aproxima-se o valor da classe y [53].

O modelo é conhecido como *feedforward* devido à direção do fluxo de dados ao longo da função que é validada para x, passando pelos cálculos de f intermediários utilizados para calcular f da última camada, obtendo a saída y [53].

Redes neurais *feedforward* são tradicionalmente denotadas a partir de uma composição de funções, de modo que cada função representa uma camada da rede. Cada uma das funções contidas na cadeia, representa uma camada da rede neural, em que as funções que se encontram entre a mais interna e a mais externa representam as camadas ocultas; a quantidade de funções é o que dá profundidade ao método. É a partir desta definição que vem o nome "aprendizado profundo". Na equação 2.11 é possível observar a abstração de uma rede neural do tipo *feedforward* [53].

$$f(x) = f^{(n)}(f^{(n-1)}(f^{(n-2}(\cdots(f^{(1)}(x))\cdots))) \tag{2.11}$$

2.7.2 Redes Neurais Convolucionais

Redes neurais convolucionais (*Convolutional Neural Networks, CNN*) ganham este nome a partir da operação linear entre matrizes, chamada de convolução. O método é comumente utilizado para resolver problemas que envolvem reconhecimento de padrões, sejam eles de voz ou imagem [58].

Para definir o motivo da convolução, é necessário assumir um problema em que uma rede neural receberia todos os *pixels* de uma imagem como dados de entrada. Desta forma, se a imagem tiver tamanho $N \times N \times 3$, será necessária uma quantidade de $3N^2 * 1$ conexões sinápticas entre a camada de entrada e a próxima camada, para um único neurônio. Se a rede neural necessitar de mais neurônios, o número de operações tende a crescer muito, de forma a ficar muito lento [58].

Para otimizar o problema, propõe-se que apenas um pequeno subconjunto (convolução) da imagem seja utilizado para o treinamento da rede neural, de forma que o tamanho deste subconjunto seja definido por M tal que $M \ll N$. A tendência é que isso diminua significativamente a quantidade de operações realizadas durante o treinamento [58].

A equação 2.12 denota como é calculada a convolução da matriz de uma imagem utilizada no treinamento de uma CNN, de forma que $S(i, j)$ é a saída da próxima camada, I representa a imagem, K representa o núcleo de convolução, e $*$ é a operação de convolução [58][53].

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(i - m, j - n)K(m, n) \tag{2.12}$$

Outras camadas que fazem parte de uma rede neural convolucional são a camada de *pooling* e a camada totalmente conectada. A função da primeira é diminuir ainda mais a complexidade do método para as próximas camadas, que é descrita em [58] como sendo uma operação semelhante a uma redução na resolução da imagem. A segunda, por sua vez, vem do mesmo conceito de redes neurais tradicionais, em que todos os neurônios da camada se conectam a todos os neurônios da camada mais à frente [58]. A figura 11 ilustra um modelo de CNN.

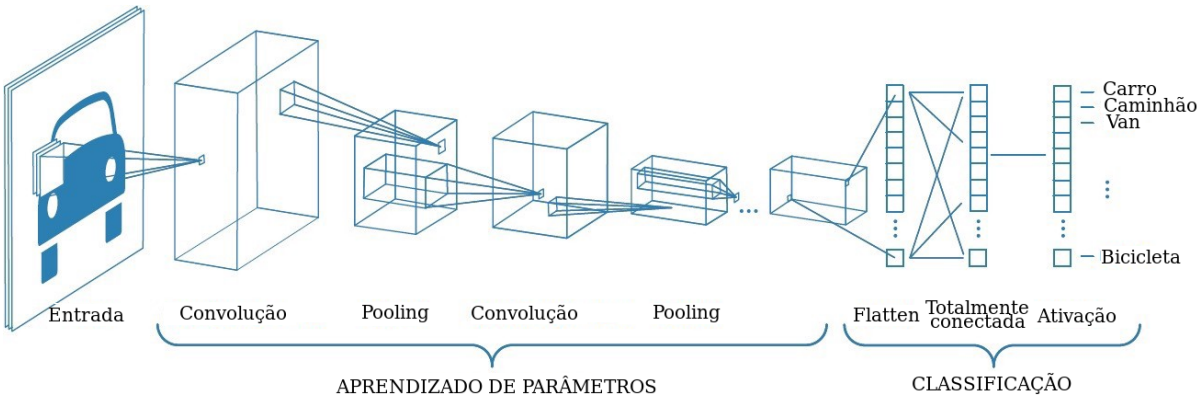


Figura 11 – Ilustração de um modelo de Rede Neural Convolucional. Fonte: adaptado de [59]

2.7.3 *Autoencoders*

Um *autoencoder* é um modelo de redes neurais profundas que faz parte da classe de algoritmos de aprendizado não supervisionado [60] e tem como intuito reduzir a dimensionalidade de um problema, a partir de mecanismos que permitam a cópia dos valores da camada de entrada para a camada de saída. O método possui uma camada oculta h que descreve um código utilizado para representar os valores de entrada. A rede neural pode ser dividida em duas partes apresentadas nos itens a seguir [53].

- Uma função de codificação $h = f(x)$;
- Uma função de decodificação que gera um reconstrutor $r = g(h)$;

Na figura 12 é possível vislumbrar a arquitetura de uma rede neural da classe dos *autoencoders*. Também é válido notar que o tamanho da camada de entrada é igual ao tamanho da camada de saída.

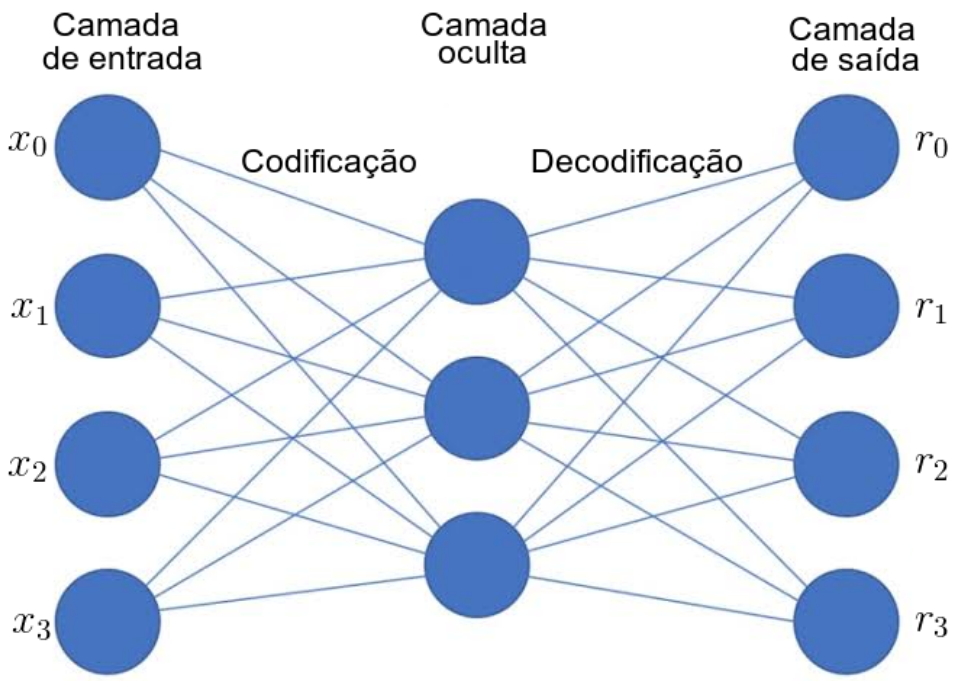


Figura 12 – Ilustração de um *autoencoder* tradicional. Fonte: adaptado de [61]

Autoencoders têm como característica realizar a codificação dos dados de entrada por meio da camada oculta e a sua decodificação pela camada de saída aplicada numa função de ativação. Desta forma, o método é construído de forma que não realize uma cópia idêntica dos valores de entrada para os valores de saída, mas que seja capaz de priorizar os aspectos mais importantes dos *inputs* para que consiga aprender as propriedades mais significativas dos dados [53]. O método é comumente tido como um caso especial das redes neurais *feedforward*, sendo assim, é comum que as mesmas técnicas utilizadas para treinamento de uma FNN sejam aplicadas aos *autoencoders* [53].

2.7.3.1 *Undercomplete Autoencoders*

A ideia de copiar os valores de entrada para a camada de saída aparenta não ter utilidade. Todavia, o processo de treinamento do *autoencoder* tem como intuito codificar os valores de entrada para que seja possível extrair as informações mais relevantes dos dados a partir da camada oculta h. Uma das maneiras de realizar essa tarefa é a partir da criação de uma camada oculta cujas dimensões sejam estritamente menores do que as dimensões da camada de entrada, da mesma forma como a figura 12 mostra. Um *autoencoder* que tem essa característica é chamado de *undercomplete* [53].

O processo de aprendizado é denotado pela minimização da função de erro

$$L(x, g(f(x))), \tag{2.13}$$

em que:
L representa a função do erro.
$h = f(x)$ é a função de codificação.
$g(h)$ é a função de reconstrução.

Um *undercomplete autoencoder* aprende o principal subespaço do conjunto de treinamento quando L representa o erro médio quadrado e quando $g(x)$ é uma função linear [53].

2.7.3.2 *Autoencoders* Esparsos

Um *autoencoder* esparso é comumente utilizado para aprender características de tarefas como classificação. O modelo possui características idênticas a um *autoencoder* tradicional, com a adição de uma função de penalidade $\Omega(h)$ na camada de codificação. Desta forma o treinamento é denotado na equação a seguir [53].

$$L(x, g(f(x))) + \Omega(h) \tag{2.14}$$

A função de penalidade pode ser entendida como um termo de regularização adicionado a uma FNN cujo objetivo é executar um aprendizado não supervisionado (cópia dos valores de entrada para os valores de saída), de forma que a dependência de $\Omega(h)$ adiciona um processo de aprendizado supervisionado ao método [53].

2.7.3.3 *Denoising Autoencoders*

Considerando a maneira que um *autoencoder* opera ao copiar os valores de entrada para os valores de saída, os *denoising autoencoders* (DAE) funcionam de uma forma um pouco diferente.

Ao invés de simplesmente mapear os valores originais de entrada para a saída a partir da minimização da função de erro, os valores de entrada são corrompidos com algum

tipo de ruído, de forma que $\tilde{x} = ruído(x)$. Assim, o treinamento ocorre ao minimizar a função de custo apresentada a seguir, indicando a ocorrência da codificação nos parâmetros $\tilde{x}$ [53].

$$L(x, g(f(\tilde{x}))) \tag{2.15}$$

2.8 Redes Neurais Recorrentes

Redes Neurais Recorrentes (*Recurrent Neural Networks*, RNN) são um gênero de redes neurais especializadas em processar sequências de valores $x^1, \cdots, x^t$ distribuídas ao longo do tempo. O método que surgiu a partir da noção de compartilhamento de parâmetros, já encontrada em aprendizado de máquina e modelos estatísticos da década de 1980 [53]. O compartilhamento de parâmetros é necessário para generalizar um modelo para sequências de diferentes tamanhos, bem como manter o peso estatístico ao longo destes tipos de sequência [53].

Modelos de Redes Neurais Recorrentes podem ser concebidos como uma espécie de grafo cíclico, em contraste com as tradicionais FNN (*Feedforward Neural Networks*) que não possuem ciclo, tal que esses ciclos representam a importância que um valor contido numa variável no tempo presente (t) tem em seu próprio valor no tempo futuro ($t+1$) [53]. Nestes recorrentes ciclos das informações, os dados de saída são alimentados de volta nas camadas anteriores do sistema. Essa característica é responsável pela principal diferença das redes recorrentes às redes neurais tradicionais do tipo *feed-forward*, pois faz com que o modelo seja capaz de sistematizar dependência entre os dados [51].

Durante o treinamento de uma RNN a informação passada pela rede não irá apenas executar o caminho que parte da camada de entrada e vai para a camada de saída, mas irá também armazenar informações dos dados em suas células, sendo que essa persistência nas informações ocorre nos ciclos presentes nos neurônios da rede. Na figura 13 é possível observar um neurônio "enrolado", representado por A, possuidor de ciclo, tendo x_t como valor de entrada e h_t como valor de saída.

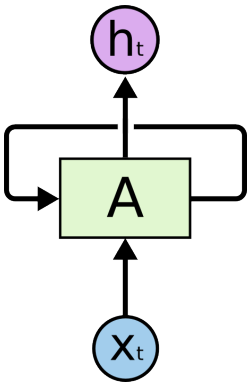


Figura 13 – Neurônio de uma RNN em seu estado "enrolado". Fonte: [62]

A quantidade de repetições destes *loops* é um processo finito definido na modelagem da rede neural, de forma que podem ser visualizados como várias cópias do neurônio de forma sequencial; processo chamado *unfolding* (desenrolar), ilustrado na figura 14, mostrando que este tipo de modelo está diretamente ligado a sequências e listas.

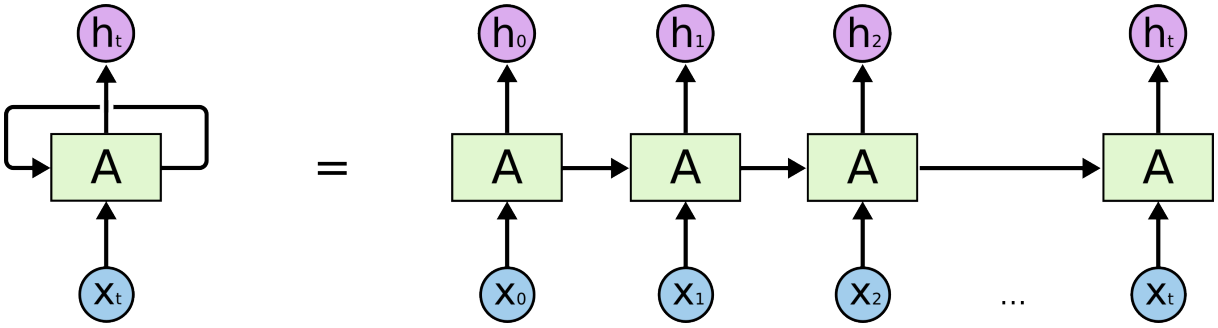


Figura 14 – Neurônio "desenrolado"de uma RNN. Fonte: [62]

Na figura 15 é apresentado um exemplo de uma célula do RNN, em que o dado de entrada x^t e a^{t-1} que é o estado oculto anterior contendo informações do passado, W_{aa} e W_{ax} representam as matrizes de peso e b_a representa os vetores de bias. Tem como saída a^t que é passada para a próxima célula e a saída $\hat{y}^t$ é aplicada em uma função de ativação, neste caso a *softmax*, e realiza a previsão para o intervalo de tempo t. Na figura 16 tem-se um caso de RNN com apenas uma camada. É possível observar a sequência de entrada $\vec{x} = x^1, \cdots, x^{T_x}$ transportada por T_x intervalos de tempo e cada uma das células possui uma previsão de saída dada por $y = y^1, \cdots, y^{T_x}$.

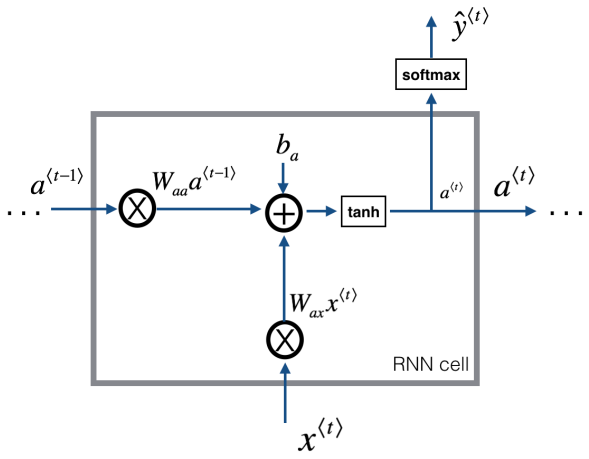


Figura 15 – Célula RNN. Fonte: [63]

A representação matemática de uma RNN é dada pela fórmula

$$a^t = \tanh(W_{ax} \cdot x^t + W_{aa} \cdot a^{t-1} + b_a) \tag{2.16}$$

e a saída é dada por

$$\hat{y}^t = g(W_{ya} \cdot a^t + b_y) \tag{2.17}$$

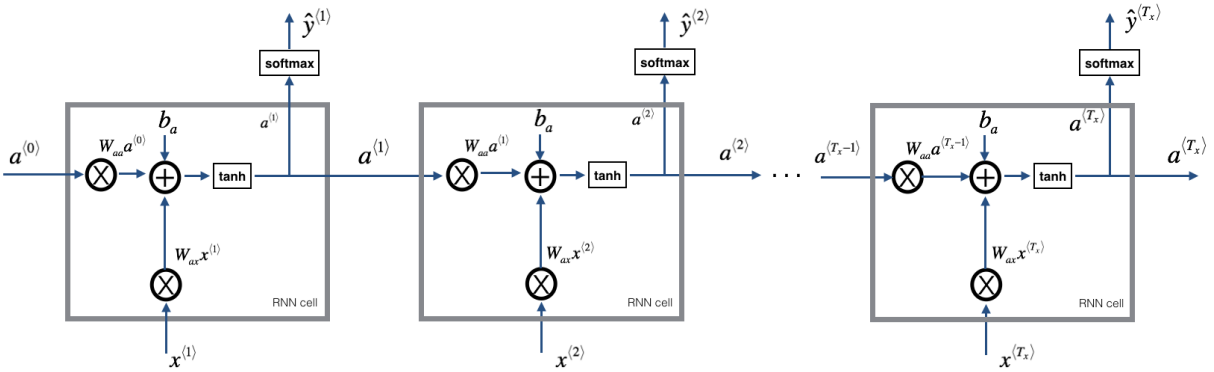


Figura 16 – Repetição da célula RNN ao longo de T_x intervalos de tempo. Fonte: [63]

2.8.1 Uso de RNNs para análise de tráfego

O uso de Redes Neurais Recorrentes para análise de tráfego tem se mostrado promissor, já que o tráfego de uma rede é tipicamente um conjunto de dados distribuídos em função do tempo, ademais, o método tem capacidade de capturar dependências temporais contidas neste tipo de informação, além de trabalhar bem com picos súbitos nos dados do tráfego. Modelos estatísticos tradicionais não são eficazes para descrever informações distribuídas de tal maneira, principalmente pela dificuldade de capturar dependências em longo prazo [51].

2.8.2 Problemas do RNN

Redes Neurais Recorrentes fazem uso de matrizes de peso W, a qual representa os pesos de cada ligação neuronal de uma camada, além disso, RNN criam estruturas bastante profundas a partir de repetidas execuções da mesma operação em uma longa cadeia temporal. A junção de ambos os fatos implica dois problemas deste tipo de modelo de rede neural, sendo eles o problema de dissipação do gradiente (*Vanishing Gradient Problem*) e o problema de explosão do gradiente (*Explosion Gradient Problem*) [53].

2.8.3 Instabilidades do Gradiente

Os problemas de instabilidade do gradiente são comumente conhecidos como dissipação e explosão do gradiente. Estes problemas ocorrem, principalmente, devido às multiplicações sucessivas que aparecem durante o treinamento da rede neural, como por exemplo no algoritmo de *backpropagation*. Na equação 2.18 é possível observar a fórmula

do método, e em 2.19, a expansão da mesma.

$$\begin{aligned} \frac{\partial C}{\partial w_{i,j}^{(l)}} &= a_k^{(l-1)} g'(z_j^{(l)}) \frac{\partial C}{\partial a_j^{(l)}} \\ \frac{\partial C}{\partial a_j^{(l)}} &= \sum_{j=0}^{n_{l+1}-1} w_{j,k}^{(l+1)} g(z_j^{(l+1)}) \frac{\partial C}{\partial a_j^{(l+1)}} \end{aligned} \tag{2.18}$$

$$\begin{aligned} \frac{\partial C}{\partial w_{i,j}^{(l)}} = a_k^{(l-1)} g'(z_j^{(l)}) \Bigg(& \sum_{j=0}^{n_{l+1}-1} w_{j,k}^{(l+1)} g'(z_j^{(l+1)}) \\ & \Bigg(\sum_{j=0}^{n_{l+2}-1} w_{j,k}^{(l+2)} g'(z_j^{(l+2)}) \cdots \\ & \Bigg(\sum_{j=0}^{n_{L-1}-1} w_{j,k}^{(L-1)} g'(z_j^{(L-1)}) \Big(2(a^{(L)} - y) \Big) \Bigg) \cdots \Bigg) \Bigg) \end{aligned} \tag{2.19}$$

Onde:
L é o número de camadas;
$n_l - 1$ é a quantidade de neurônios na camada l;
C é a função de custo;
$w_{j,k}^{(l)}$ é o peso de uma conexão que liga o nó k da camada $l-1$ ao nó j da camada l;
$z_j^{(l)}$ é o valor de entrada para o nó j na camada l;
$a_j^{(l)}$ é o valor de saída do nó j na camada l;
$g(\cdot)$ é a função de ativação;

2.8.3.1 Dissipação e explosão do gradiente e o problema de dependências a longo prazo

Seja a rede neural denotada pela figura 17, com L camadas, tal que L é um número muito grande. Assim, considerando o cálculo do *backpropagation* para um nó qualquer de uma camada, quanto mais próxima ela estiver da camada de entrada, maior serão as quantidades de multiplicações necessárias para atualizar o valor de seu peso, portanto, se o valor dos produtos dos somatórios internos do cálculo do *backpropagation* resultarem em valores estritamente menores do que 1, o resultado final será um número muito pequeno que, por consequência, não causará nenhuma mudança significativa no peso sendo analisado [64][65]. E é devido a este fato que Redes Neurais Recorrentes possuem dificuldade em manter dependências a longo prazo, afinal camadas muito distantes tendem a perder a influência que exercem entre si, devido ao problema de dissipação do gradiente [65][66].

O mesmo ocorre para a explosão do gradiente, porém, os valores dos produtos internos do *backpropagation* agora são estritamente maiores do que 1, podendo implicar

os resultados das multiplicações sucessivas desses somatórios em um número muito maior do que 1, fazendo com que o peso sináptico se afaste do valor ótimo [67].

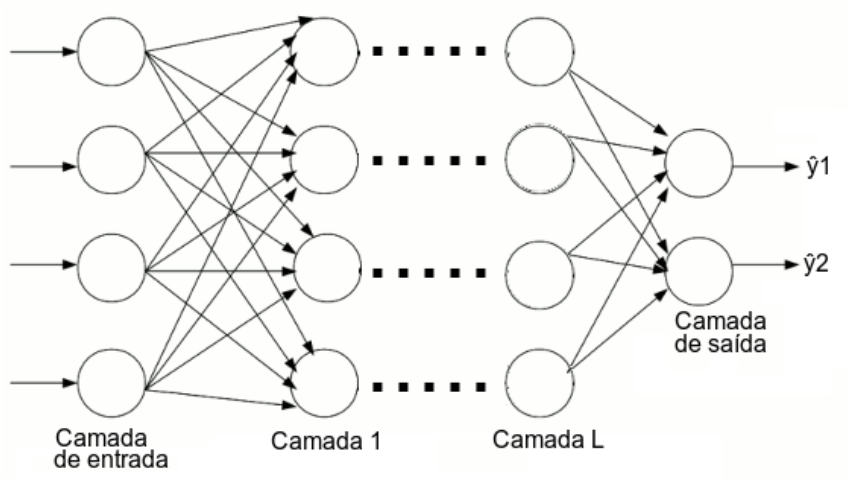


Figura 17 – Rede neural de L camadas, utilizada para ilustrar os problemas de instabilidade do gradiente. Fonte: próprio autor.

Diante da ocorrência do problema de dissipação do gradiente, fica difícil saber qual caminho tomar para melhorar a função de custo, ou seja, é quando o valor do gradiente fica pequeno o bastante para se tornar insignificante no aprendizado da rede e impedir a captação de dependências a longo prazo. O problema de explosão do gradiente faz com que o aprendizado se torne instável [53].

2.9 Long Short-Term Memory

O LSTM (*Long Short-Term Memory*) é uma evolução do RNN. O método tem como premissa evitar o problema de dependências a longo prazo do RNN clássico, que consiste na falha em capturar as dependências a longo prazo contidas nas informações sendo processadas [68][69], a partir da correção das falhas de dissipação e explosão do gradiente[70]. O problema é amplamente discutido por Bengio, et al. [65] que define dependências a longo prazo como sendo uma tarefa cuja saída desejada no período de tempo t depende de uma entrada apresentada em um período de tempo anterior τ, em que $\tau \ll t$.

2.9.1 Como funciona o LSTM

O LSTM funciona de maneira semelhante ao RNN, estruturado na forma de cadeia em que cada camada representa um intervalo de tempo t como ilustra a figura 18. Durante a explicação a seguir, os valores de W e b com índices subscritos, presentes nas equações 2.20, 2.21 e 2.23 indicam, respectivamente, as matrizes de peso e os valores de *bias*.

O primeiro passo do método é decidir se a informação do estado anterior será removida do estado atual da célula. Esta operação é executada pela camada *forget gate*,

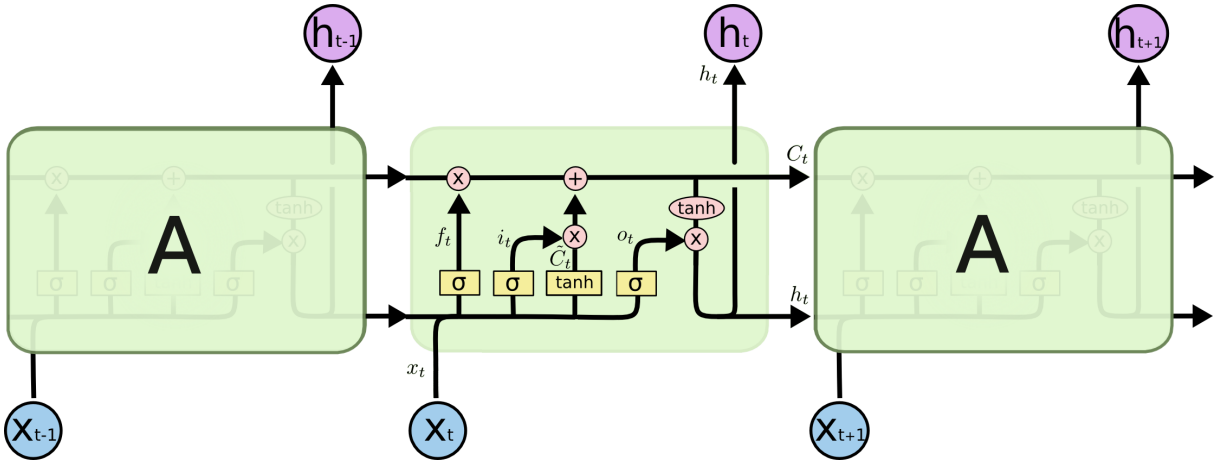


Figura 18 – Cadeia LSTM. Fonte: adaptado de [62]

representado na equação por f_t, em que produz como saída valores entre 0 (esquecer) e 1 (manter). Esta camada é representada pela seguinte equação:

$$f_t = sigmoid(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.20}$$

O segundo passo decide qual informação será armazenada no estado da célula. Composto por uma primeira camada chamada de *update gate*, denotada por i_t, que decide qual valor será atualizado, e uma segunda que calcula o valor candidato, representado por $\tilde{C}_t$, que pode ser adicionado ao estado atual da célula, substituindo o estado anterior. Cada uma das partes é representada pelas seguintes equações:

$$\begin{aligned} i_t &= sigmoid(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \end{aligned} \tag{2.21}$$

Em seguida, o estado da célula, representado por C_t, será atualizado utilizando as informações produzidas anteriormente, levantando a possibilidade do estado anterior continuar na memória ou ser descartado. A equação é formada pela multiplicação entre o *forget gate* do passo atual (t) e o valor memorizado pelo passo anterior $(t-1)$ da rede neural, indicando a possibilidade de descarte do mesmo; em seguida, soma-se o resultado da operação anterior com a multiplicação do *update gate* com o valor $\tilde{C}_t$ candidato a substituir o valor memorizado pelo passo anterior. O resultado obtido em C_t será passado a diante.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{2.22}$$

Por último, é necessário computar qual será a saída da célula. Neste passo, multiplica-se o resultado do estado da célula aplicado numa função de ativação *tanh* pela saída do

output gate. Neste passo, o valor o_t representa o *output gate* e h_t representa o valor de saída do neurônio.

$$\begin{aligned} o_t &= sigmoid(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t \cdot \tanh(C_t) \end{aligned} \tag{2.23}$$

2.9.2 Modelos para previsão de sequências

Existem quatro modelos primários que podem ser aplicados numa rede neural recorrente, e são utilizados para realizar previsões em sequências [71]. A nomenclatura utilizada nas figuras 19, 20, 21 e 22 é apresentada nos itens seguir:

- X: Os valores da sequência de entrada;
- u: O valor da camada oculta;
- y: A sequência de saída.

Cada um destes valores podendo ser delimitados por um passo de tempo t.

2.9.2.1 Modelo um-para-um

Um modelo um-para-um (*one-to-one*) produz como saída um valor y_t para cada valor de entrada X_t, sendo apropriado para prever um único passo, dado um passo de tempo como entrada, como mostra a figura 19 [71].

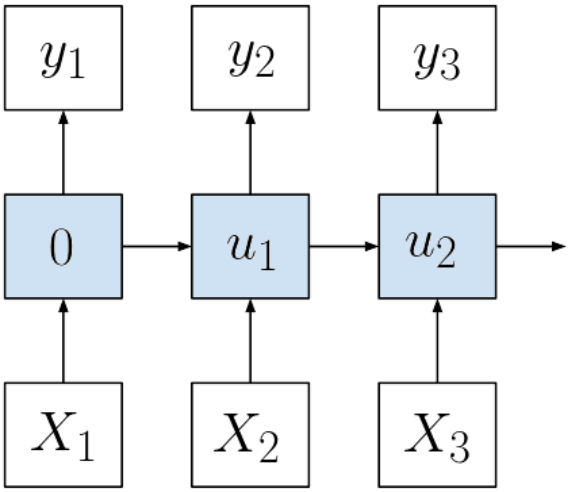


Figura 19 – Modelo um-para-um. Fonte: adaptado de [71]

Alguns exemplos de problemas solucionados por esse modelo são: previsão do próximo valor real de uma sequência numérica temporal e previsão da próxima palavra em uma frase.

2.9.2.2 Modelo um-para-muitos

Um modelo um-para-muitos (*one-to-many*) produz múltiplos valores de saída ($y_t, y_{t+1}, \cdots$), para um único valor de entrada (X_t), como mostra a figura 20. Este método é apropriado para realizar previsões que envolvam uma sequência como saída para cada passo de tempo [71].

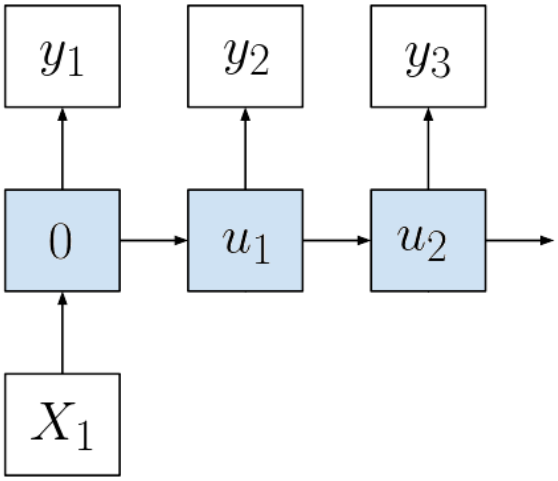


Figura 20 – Modelo um-para-muitos. Fonte: adaptado de [71]

Problemas de geração de legendas para imagem são bons exemplos para este tipo de modelo, pois dada uma imagem como entrada, a saída deve ser uma sequência de palavras que a descreve.

2.9.2.3 Modelo muitos-para-um

O modelo muitos-para-um (*many-to-one*) gera como saída um único valor (y_t) para uma sequência de entrada ($X_t, X_{t+1}, \cdots$), como ilustra a figura 21. Este modelo resolve problemas que envolvem prever o próximo valor de saída de uma sequência dada uma sequência de observações, como análise de sentimentos, que envolve, a partir da observação de uma sequência de palavras, produzir como saída um valor que rotula uma emoção [71].

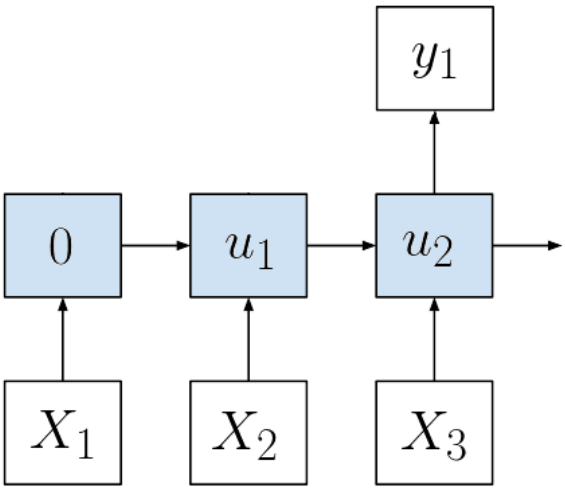


Figura 21 – Modelo muitos-para-um. Fonte: adaptado de [71]

2.9.2.4 Modelo muitos-para-muitos

Um modelo muitos-para-muitos (*many-to-many*) produz múltiplos valores de saída $(y_t, y_{t+1}, \cdots)$ a partir de múltiplos valores de entrada $(X_t, X_{t+1}, \cdots)$, processo denotado pela figura 22. É válido ressaltar que o tamanho da sequência de entrada não seja idêntico ao da sequência de saída [71].

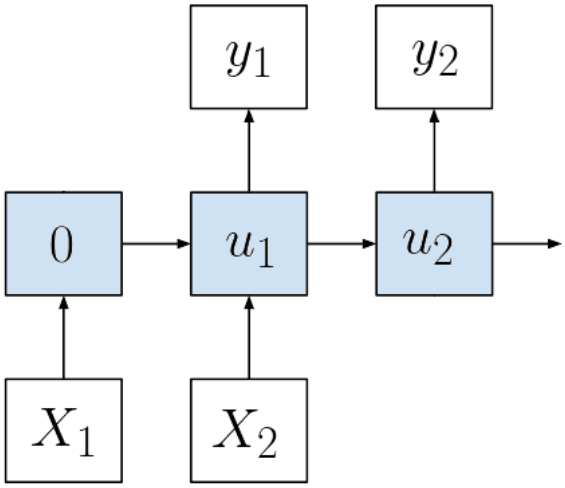


Figura 22 – Modelo muitos-para-muitos. Fonte: adaptado de [71]

Este modelo é utilizado para solucionar problemas que envolvem a sintetização de uma sequência de saída a partir de uma sequência de entrada. Um exemplo disso é tradução simultânea e classificação de áudio em palavras, que envolve como entrada uma sequência de áudio e a saída uma sequência de palavras.

2.10 Gated Recurrent Unit

A metodologia GRU faz parte da classe das redes neurais recorrentes e foi introduzida por Cho, et al.[72]. O modelo tem como objetivo corrigir o problema de dissipação e explosão do gradiente, ao mesmo tempo em que é mais eficiente que o LSTM no quesito de quantidade de cálculos realizados pelo método, isto é, existe uma diminuição na quantidade de operações realizadas pela GRU com relação ao LSTM.

2.10.1 Funcionamento de uma GRU

O GRU funciona de maneira semelhante ao LSTM, com a diferença de que faz uso de menos operações matriciais durante o processo de aprendizado. Esta característica se da a partir da combinação do *forget gate* com o *input gate*.

O método tem esse nome devido aos valores de z e r, pertencentes ao intervalo $[0, 1]$, serem chamados de *gates* em que o primeiro representa o *update gate* e o segundo calcula a relevância do valor computado pelo passo anterior.

A figura 23 ilustra uma célula GRU. O processo de funcionamento do método se dá a partir da realização das seguintes operações matemáticas:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{2.24}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \tag{2.25}$$

As equações 2.24 e 2.25 representam os *gates*; a equação 2.24 indica o *update gate*, que ao produzir valores no intervalo $[0, 1]$ indica quando um dado memorizado deve ser mantido ou atualizado.

Caso produza o valor 0, indica que a informação deve ser mantida, entretanto, se produzir o valor 1, indica que a informação deve ser substituída. A equação 2.25 indica a relevância que o valor h_{t-1} possui ao calcular o valor candidato a substituir h_t

$$\tilde{h}_t = \tanh(W \cdot [r_t \cdot h_{t-1}, x_t]) \tag{2.26}$$

O resultado de $\tilde{h}_t$, obtido pela equação 2.26 representa o valor candidato a substituir o valor da célula h_t.

$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t \tag{2.27}$$

Por fim, a equação 2.27 indica o valor atual da célula, calculado a partir da relevância do valor candidato $\tilde{h}_t$ e o valor do tempo anterior, h_{t-1}.

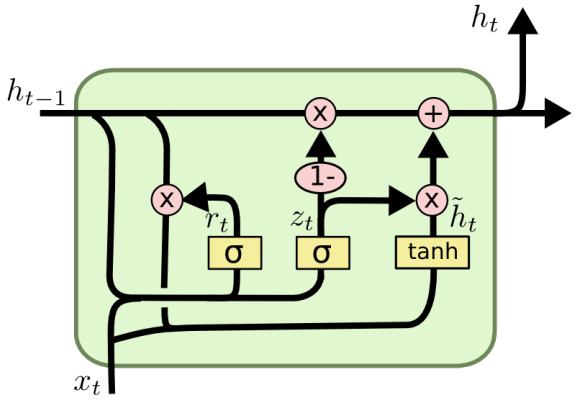


Figura 23 – Ilustração de uma célula *Gated Recurrent Unit*. Fonte: [62]

2.11 Considerações sobre o capítulo

Existem diversos modelos de *Deep Learning* que podem ser aplicados no aprendizado de padrões de dados, de forma que podem ser utilizados no processo de classificação de tráfego e detecção de anomalias. A tabela 1 apresenta um resumo dos métodos de aprendizado profundo estudados.

Tabela 1 – Resumo dos métodos de aprendizado profundo estudados.

Método	**Funcionamento**
Feedforward Neural Network	É o modelo mais comum de redes neurais profundas. Tem como objetivo aproximar o valor de uma função $f(x)$ dado um valor de entrada e um valor de saída.
Redes Neurais Convolucionais	Operam num conjunto de dados de duas dimensões. A partir disto, aplicam uma função de convolução na matriz de entrada com o objetivo de otimizar o treinamento. Este tipo de rede neural é comumente aplicado no reconhecimento de imagens.
Autoencoders	Tem como objetivo realizar a redução da dimensionalidade de um problema partindo da ideia de realizar uma cópia dos valores da camada de entrada para a camada de saída.
Redes Neurais Recorrentes	Operam com dados dispostos sequencialmente. Possuem mecanismo semelhante a um *loop*, cujo objetivo é criar dependências a longo prazo nos dados de treinamento.
Long Short-Term Memory	Variação das redes neurais recorrentes tradicionais. Possuem a mesma estrutura, porém são mais robustas e corrigem os problemas de instabilidade do gradiente que ocorriam no método tradicional.
Gated Recurrent Unit	Variação do método LSTM, cujo objetivo é simplificar o mesmo ao diminuir a quantidade de cálculos realizados por uma célula a partir da combinação de algumas equações. O método GRU também resolve os problemas de dissipação e explosão do gradiente.

3 TRABALHOS CORRELATOS

Segurança da Informação e métodos de detecção de anomalias em redes têm sido amplamente pesquisados nos últimos anos devido ao aumento do uso de dispositivos interconectados. Pesquisas nessa área tornaram-se cada vez mais importantes ao longo dos anos devido principalmente ao fato de que em paralelo ao crescimento da utilização da internet, houve também a evolução de métodos utilizados para realizar ataques em redes.

O aprendizado de máquina convencional normalmente faz uso de análise estatística para resolução de problemas; sua performance é correlacionada com o pré-processamento das informações de entrada, realizado por um processo de engenharia de dados. O aprendizado profundo é uma subclasse da inteligência artificial e, por sua vez, tem como característica usar grande quantidade de dados, precisando de pouco ou nenhum pré-processamento, além de possuir uma maior invariância a mudanças consideradas irrelevantes para o aprendizado e, ainda assim, permanecer sensível a detalhes importantes [55]. A técnica tem se tornado cada vez mais relevante, sendo utilizada mais amplamente na criação de Sistemas de Detecção de Intrusão (*Intrusion Detection System*, IDS). Moustafa et al. [73] apresentam, a partir de um levantamento bibliográfico, como diferentes métodos de *Machile Leaning* e *Deep Learning* podem ser aplicados na criação de um IDS, bem como critérios para validação do sistema.

O campo de *Deep Learning* abrange diferentes técnicas, entre elas Redes Neurais Recorrentes. Uma Rede Neural Recorrente (*Recurrent Neural Network, RNN*) é uma arquitetura de rede neural profunda cuja principal característica é operar com dados sequenciais e memorizar dependências a longo prazo em informações [53][51].

N. Ramakrishnan e T. Soni [51] abordam a previsão de tráfego como um problema de previsão de série temporal, fazendo uso de arquiteturas RNN. Os autores concluem que as técnicas são promissoras para realizar estimativas em tráfego de rede, tendo performance melhor do que métodos estatísticos tradicionais de previsão, além de serem capazes de inferir tráfego futuro com eficiência.

R. Vinayakumar et al. [70] fazem um levantamento de técnicas de *Deep Learning* utilizadas para realizar previsões em tráfego de rede; os procedimentos avaliados são o RNN e suas subcategorias. A conclusão obtida é de que o LSTM (*Long Short-Term Memory*), uma subclasse do RNN, tem uma melhor performance com relação às técnicas comuns de FFN (*Feedforward Neural Network*) e RNN.

Nos trabalhos [74], [68] e [75] é feito o uso da técnica de LSTM para detecção de anomalias e em todos eles os autores argumentam sobre a necessidade de manter a depen-

dência dos dados durante a análise, bem como a resolução do problema de dependências a longo prazo, que é resolvido pelo LSTM, citado principalmente pelos trabalhos [74] e [68].

Em [76], os autores combinam dois métodos de *Deep Learning*, o LSTM e o CNN, voltada para gerência de redes e qualidade de serviço a partir da criação de um classificador de tráfego. O método apresentado faz parte da classe de aprendizado supervisionado baseado em estatísticas, que seja capaz de detectar um serviço sendo utilizado em um fluxo IP, empregando extração de características dos cabeçalhos dos pacotes de dados comutados ao longo do período de duração de um fluxo. Para o uso do primeiro modelo, os dados são organizados na forma de uma matriz contendo uma dimensão temporal e um vetor de atributos. Já no segundo modelo, os dados utilizados para a classificação do tráfego são tratados como se fossem uma imagem, para que seja executado o processo de convolução. No fim, conclui-se que a combinação de ambos os métodos possui a melhor performance de classificação, entretanto, foi o LSTM quem mais contribuiu para os resultados obtidos.

Proença et al. [34] argumentam a necessidade de trabalhar com caracterização de tráfego a partir da análise de fluxos de rede (*Digital Signature of Network Segment Using Flow Analysis*, DSNSF) e propõem, para o cálculo de uma previsão de tráfego, o método da colônia de formigas, o método de previsão de Holt-Winters e Análise de Componentes Principais (*Principal Component Analysis*, PCA). Apesar dos três métodos terem apresentado divergência em eficácia, foi concluído que os DSNSF gerados são eficientes para descrever o comportamento normal de um tráfego de rede.

No trabalho [77], os autores criam um modelo não supervisionado de *deep learning* em duas etapas (*Two-Stage Deep Learning, TSDL*), baseado em um *auto-encoder* profundo (*Deep Stacked Auto-Encoder, DSAE*) com um classificador *softmax*. O modelo possui duas fases, sendo que a primeira delas é encarregada de classificar tráfego de redes como normal ou anormal e a segunda realiza a detecção de ataques, fazendo o uso do resultado da primeira como sendo uma *feature* extra. Por fim, conclui-se que o modelo é capaz de realizar previsões muito boas, além de possuir um tempo de execução bastante baixo.

Lin et al. [78] argumentam que modelos de aprendizado profundo são capazes de superar os modelos clássicos aplicados na detecção de anomalias em redes. Sendo assim, os autores propõem um algoritmo de redes neurais convolucionais que seja capaz de categorizar ameaças cibernéticas de maneira eficiente a partir da análise de pacotes em fluxos de rede. A partir da extração de características de um tráfego de redes, cria-se uma matriz de imagem para que a rede neural seja treinada. Na fase de validação, observaram que o erro de classificação diminuía inversamente proporcional ao tamanho do conjunto de testes. A acurácia obtida foi de mais de 97%.

Devido aos métodos de aprendizado profundo terem se mostrado promissores, Naseer et al. [79] desenvolveram modelos de detecção de anomalias baseados em diferentes estruturas de redes neurais profundas, tais como CNN, *auto-encoders* e RNN, e compararam a acurácia destes modelos com modelos convencionais de aprendizado de máquina. Todos os modelos de redes neurais profundas obtiveram bons resultados, sendo que os melhores valores foram produzidos pelo modelo de LSTM, deixando o CNN em segundo lugar e os métodos convencionais aparecendo em seguida.

No trabalho de Zhao et al. [80] os autores têm como foco os problemas existentes em métodos de detecção de intrusão que fazem uso de redes neurais, sendo eles: informações redundantes, grande quantidade de dados, tempo grande de treinamento, e facilidade de cair em ótimo local. Os autores também fazem a proposta de um método que utiliza uma rede de crença profunda (*Deep Belief Network, DBN*) e rede neural probabilística (*Probabilistic Neural Network, PNN*). Outro procedimento realizado neste trabalho é o de otimização do aprendizado da rede a partir do uso de Otimização por Enxame de Partículas (*Particle Swarm Optimization, PSO*). Por fim concluem, a partir dos resultados experimentais, que a combinação de *deep learning*, PSO e PNN é efetiva e consegue solucionar os problemas supracitados, no campo de detecção de intrusões.

Modelos lineares de previsão de tráfego de redes não funcionam tão bem devido à não linearidade do fluxo de informações. Assim, Fu et al. [81] realizaram uma pesquisa com o intuito de proteger sistemas de transporte inteligente, que fazem parte do conceito de cidades inteligentes. Para isso, são propostos os modelos LSTM e GRU de aprendizado profundo para realizar previsão em tráfego. O argumento utilizado pelos autores ao utilizar as metodologias propostas é de que, para lidar com fluxo de redes, é necessário um método de *Deep Learning* capaz de lidar bem com séries temporais. Desta forma, são citados no artigo os métodos de LSTM e GRU como possuidores de boa performance neste tipo de situação. Desta forma, a partir da utilização dos dois modelos de redes neurais recorrentes citados, os autores tentam encontrar qual das duas metodologias produzirá a melhor performance. Para isso, foi realizada a comparação do MAE entre um método estatístico chamado de modelo auto-regressivo integrado de médias móveis (*Autoregressive Integrated Moving Average, ARIMA*), LSTM e GRU e a conclusão foi de que os modelos de aprendizado profundo testados superam os modelos ARIMA para previsão de tráfego. Além disso, o modelo GRU implementado foi capaz de reduzir o valor do erro 5% a mais do que o modelo LSTM.

Yuan et al. [82] propõem um sistema chamado *DeepDefense*, que utiliza métodos de *deep learning* para detecção de DDoS. Os modelos implementados que fazem parte do sistema são o CNN, o LSTM e o GRU, sendo que os dois últimos são modelos de redes neurais recorrentes. O objetivo do uso de ambos os modelos foi que o LSTM consegue superar o problema de dissipação do gradiente que ocorre no RNN tradicional, e

o GRU é uma variação mais simples do LSTM, que pode ser treinado mais rapidamente devido à menor quantidade de parâmetros. Ainda neste trabalho, os autores comparam os modelos propostos de redes neurais recorrentes entre si, e também comparam com o método *Random Forest*, caracterizado como *shallow learning*. Os resultados experimentais demonstraram que o sistema foi capaz de reduzir significativamente a taxa de erro em relação a métodos tradicionais de aprendizado de máquina, além de mostrar que os métodos de RNN foram capazes de aprender padrões de tráfego a longo prazo e generalizar melhor do que o método *Random Forest*. Entretanto, quando os modelos de redes neurais recorrentes foram comparados entre si, a diferença de performance produzida foi muito baixa, indicando que os métodos obtiveram resultados satisfatórios. Além disso, o método CNN não foi capaz de assimilar características uteis, devido à falta de atributos de teor espacial no problema, e, portanto, não conseguiu obter resultados melhores que os métodos de LSTM e GRU.

R. Bipraneel e C. Hon [83] fizeram o uso de LSTM bi-direcional (*Bi-Direction Long Short-Term Memory, BLSTM*) com o objetivo de criar um IDS para detectar ataques em ambiente de internet das coisas (*Internet of Things, IoT*) a partir de uma classificação binária de padrões normais e padrões de ataque encontrados no tráfego da rede. Para a implementação do modelo, os autores fazem uso da linguagem *Python*, juntamente com as bibliotecas *Tensorflow* e *Keras*, voltadas para *machine learning*.

Os ataques estudados no decorrer do artigo são os ataques de Análise, que visam obter informações da rede a partir do farejamento de pacotes ou escaneamento de portas, *Backdoor*, Negação de Serviço, *Worms* e Reconhecimento, sendo um termo mais abrangente para se referir a ataques que fazem qualquer tipo de mapeamento ilegítimo na rede.

O estudo conclui que o modelo fundamentado em BLSTM foi capaz de aprender características detalhadas da base de dados durante o processo de treinamento.

No trabalho de Usterbay et al. [84] o objetivo é determinar os efeitos das características dos dados em encontrar as que possuem maior importância para o conjunto. Para isso, o método que os autores utilizam é o de eliminação recursiva de características (*Recursive Feature Elimination, RFE*) através do algoritmo de floresta aleatória.

A partir disso, o sistema proposto pelos autores faz com que os dados de entrada da rede neural passem primeiramente pela eliminação de características, para depois serem alimentados ao modelo de *deep learning* utilizado. Como consequência da aplicação do RFE, foi concluído que um número menor de características, dependendo do conjunto de dados, não reduz a acurácia do método, embora seja capaz de produzir melhorias significativas na velocidade do mesmo.

Em [85], os autores realizam a implementação de um modelo de *autoencoder* tradi-

cional e *autoencoder* convolutivo para detecção de anomalias. O *autoencoder* convolutivo desempenha uma redução da quantidade de parâmetros com relação ao *autoencoder* tradicional, implicando em um menor tempo de treinamento. Segundo os autores, a escolha do método se da a partir da importância de identificar a não linearidade presente nas características do tráfego de rede, tendo como finalidade obter a correlação não linear existente entre elas. Isso se dá a partir da redução da dimensionalidade no *dataset*, com o objetivo de encontrar um subespaço ótimo capaz de diferenciar tráfego normal de tráfego anômalo. Por fim, os autores concluem que, a partir do uso de *autoencoder* foi possível capturar facilmente a correlação entre as características do *dataset*. Além disso, o método convolutivo produziu resultados melhores do que outros métodos de detecção.

No trabalho de Al-Qatf et al. [86] foi apresentado um sistema composto pela combinação entre *autoencoders* esparsos e Máquinas de Vetores de Suporte (*Support Vector Machines*, SVM) para a criação de um sistema de detecção de intrusão em redes (*Network Intrusion Detection System*, NIDS). O modelo deste trabalho tem como características o aprendizado não supervisionado a partir do uso do *framework self-taught learning (STL)*. A abordagem do problema se dá de forma que ocorra aprendizado de características e redução na dimensionalidade, de forma que há uma redução no tempo de treinamento e um aumento da acurácia da SVM no que se refere à detecção de anomalias. Os autores comparam seus resultados com os de trabalhos correlatos e concluíram que o modelo que implementaram apresenta uma melhoria na acurácia da classificação da SVM, bem como a aceleração do processo de treino e teste.

3.0.1 Considerações Finais Sobre o Capítulo

A partir da análise dos diversos modelos de redes neurais utilizados pelos autores dos trabalhos apresentados, foi decidido o método de redes neurais profundas utilizado neste trabalho, o *Long Short-Term Memory*. O uso da técnica foi motivada, principalmente, a partir da análise dos trabalhos desenvolvidos por N. Ramakrishnan e T. Soni [51], que utilizam um modelo RNN tradicional para detecção de anomalias em redes, e dos trabalhos conseguintes, que fazem uso do LSTM como foco principal da pesquisa, sendo eles os trabalhos de A. H. Mirza e S. Cosan [75], S. A. Althubiti et al. [68] e G. Qin et al. [74]. Além disso, também foram considerados os trabalhos que utilizaram de mais de um modelo a fim de compará-los com outras metodologias de *deep learning* no âmbito de detecção de anomalias em redes, sendo eles os trabalhos realizados por R. Bipraneel e C. Hon [83], Yuan et al. [82] e Fu et al. [81].

4 DESENVOLVIMENTO

A capacidade dos métodos LSTM e GRU de realizar previsões em séries temporais pode ser aplicada na resolução dos mais variados tipos de problemas, assim como na segurança da informação. Neste trabalho, a aplicação da técnica será na previsão de tráfego de rede, pois o mesmo se comporta como uma série temporal. A ideia central será a de, a partir do treinamento da rede neural, gerar 24 horas de um tráfego normal de rede, sem a existência de anomalias, e compará-lo com um tráfego real para, assim, averiguar a existência de uma anomalia. Este procedimento será executado para os métodos LSTM e GRU e, por fim, será feita uma comparação dos resultados obtidos por ambos.

A implementação deste sistema se deu na linguagem *Python*, que é uma linguagem interpretada, interativa e orientada a objetos [87] e que tem se tornado uma das mais populares linguagens de programação quando o assunto é *machine learning* e *data science* [88]. As bibliotecas utilizadas para processamento e manipulação dos dados foram *Numpy*, *Pandas* e *Scikit-Learn*. Para geração e treinamento da rede neural, foi utilizado a biblioteca *Keras*, que é uma API de alto nível para redes neurais, escrita em *Python* e capaz de funcionar sob a biblioteca *Tensorflow* [89].

4.1 Dados

A coleta dos dados utilizados no treinamento do modelo foram feitas a partir de uma rede SDN criada no *Mininet*, juntamente com o controlador *Floodlight* e os fluxos IP obtidos do *OpenFlow* [90]. Os atributos coletados foram: quantidade de *bytes*; valores de IP de origem; valores de IP de destino; valores de porta de origem; valores de porta de destino e quantidade de pacotes.

Após a coleta destes dados, os mesmos foram armazenados num arquivo de texto em formato *csv*; os atributos foram distribuídos em colunas e cada linha do arquivo representa uma fração de segundo da coleta, de forma que as amostras de cada uma das dimensões continha dados dentro de intervalos menores do que 1 segundo, totalizando 24 horas. O local de coleta foi feito no laboratório do grupo de redes *Orion* do Departamento de Computação da Universidade Estadual de Londrina.

Por fim, cada atributo foi distribuído em um arquivo separado para si, e suas frações de segundo, quando menores do que 1 segundo, foram agrupadas em uma única linha, de forma a representar um único segundo, bem como os cálculos das entropias para as dimensões contendo os valores de IP de origem, valores porta de origem, valores de IP de destino e valores de porta de destino; gerando, assim, 6 arquivos que representam cada uma das dimensões, tal que cada arquivo possui 86400 linhas, sendo a quantidade total

de segundos contidos em 24 horas. Cada dimensão foi nomeada como mostra a Tabela 2 e os arquivos de tráfego são representados pela Tabela 3.

Tabela 2 – Nomes atribuídos a cada uma das dimensões coletadas.

Descrição do atributo	**Nome concedido**
Quantidade de *bytes*	*Bytes*
Entropia de IP de destino	H(dstIP)
Entropia de portas de destino	H(dstPort)
Entropia de IP de origem	H(srcIP)
Entropia de portas de origem	H(srcPort)
Quantidade de pacotes	Pacotes

Tabela 3 – Representação de cada um dos arquivos contendo dados de tráfego.

t	*Bytes*
0	6928.0
1	28524.0
2	58675.0
3	93526.0
4	127653.0
5	47400.0
6	45362.0
⋮	⋮
86398	84432.0
86399	67340.0

t	H(dstIP)
0	3.7216117
1	6.0164078
2	6.0164078
3	6.0367901
4	6.0175594
5	5.1565647
6	5.5045078
⋮	⋮
86398	5.9743301
86399	6.1108154

t	H(dstPort)
0	3.1105780
1	3.8098595
2	3.8098595
3	3.7937935
4	3.8000918
5	3.4622673
6	3.7724653
⋮	⋮
86398	3.9340112
86399	3.9947518

t	H(srcIP)
0	4.2479275
1	6.6865005
2	6.6865005
3	6.7004397
4	6.7004397
5	5.7402239
6	5.9307373
⋮	⋮
86398	6.5235619
86399	6.6724253

t	H(srcPort)
0	3.4713544
1	4.2479831
2	4.2479831
3	4.2344043
4	4.2344043
5	4.1253263
6	4.0884333
⋮	⋮
86398	4.0369778
86399	4.1585342

t	Pacotes
0	34.0
1	147.0
2	300.0
3	478.0
4	659.0
5	240.0
6	236.0
⋮	⋮
86398	421.0
86399	345.0

As dimensões escolhidas para fazer a previsão foram as de entropia de IP de destino e entropia de porta de destino. O motivo da escolha destas dimensões se deu a partir da análise do comportamento das mesmas durante um dia inteiro de tráfego sem anomalias e um dia inteiro de tráfego contendo a anomalia de DDoS. Como é possível observar na figura 24, as dimensões que possuem comportamento mais suave são as dimensões de

entropia, assim como na figura 25 é possível observar que a maior discrepância nos dados ocorre nas dimensões de entropia de IP de destino e entropia de portas de destino; isso ocorre devido ao fato de que um ataque DDoS direciona o tráfego de diversos dispositivos para um único ponto da rede.

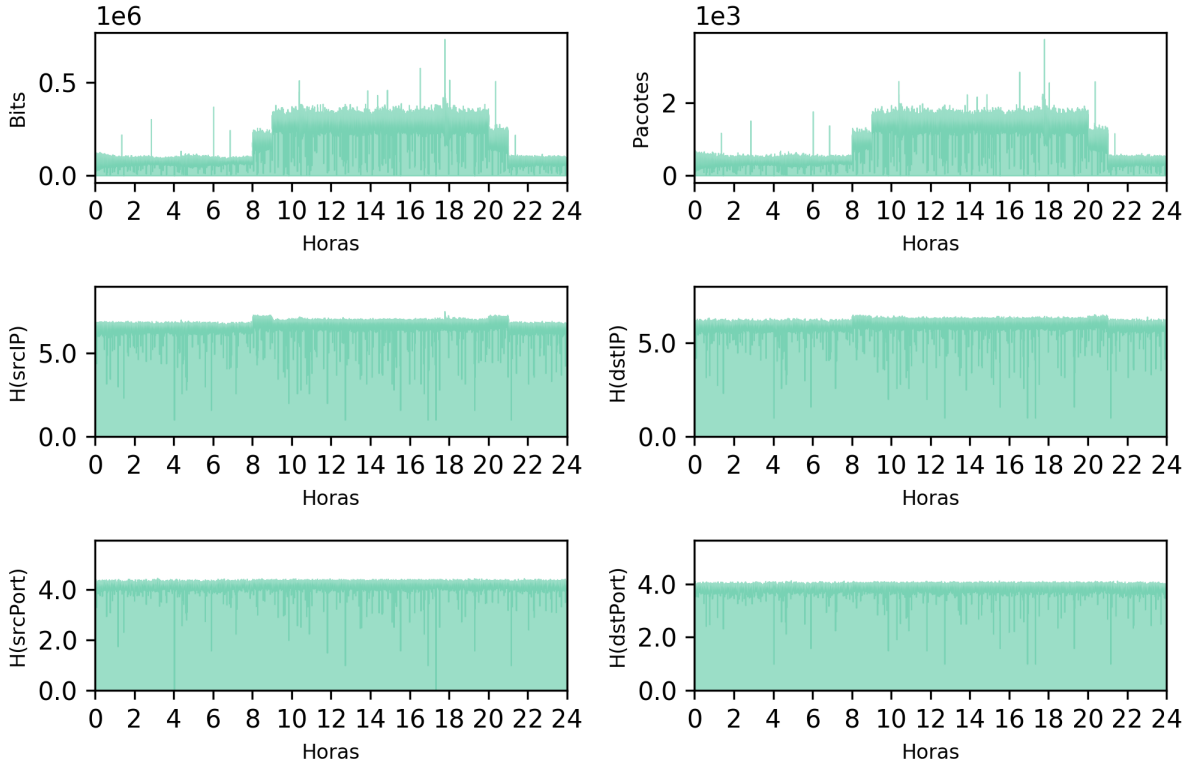


Figura 24 – Representação gráfica de cada um dos atributos coletados de um fluxo de rede, sem a ocorrência de ataque. Fonte: próprio autor.

4.2 Métricas utilizadas na avaliação do tráfego previsto

A seguir serão apresentadas algumas métricas que podem ser utilizadas para calcular o valor de dispersão entre um conjunto de predição e um conjunto real de dados. Neste caso, o conjunto real será representado pelo conjunto de testes retirado da base de dados de treinamento da rede neural LSTM e GRU, de forma que este conjunto representa um dia de tráfego real. O conjunto de previsão é representado pelo tráfego de saída da rede neural, também sendo um conjunto que representa um dia de dados.

4.2.1 Acurácia

Uma métrica importante para o desenvolvimento deste trabalho é a métrica de acurácia, que será utilizada para obter a melhor configuração de treinamento da rede neural. Para definir esta métrica é necessário assumir a existência de dois vetores: um vetor contendo os valores gerados pela rede neural e um vetor contendo os valores esperados.

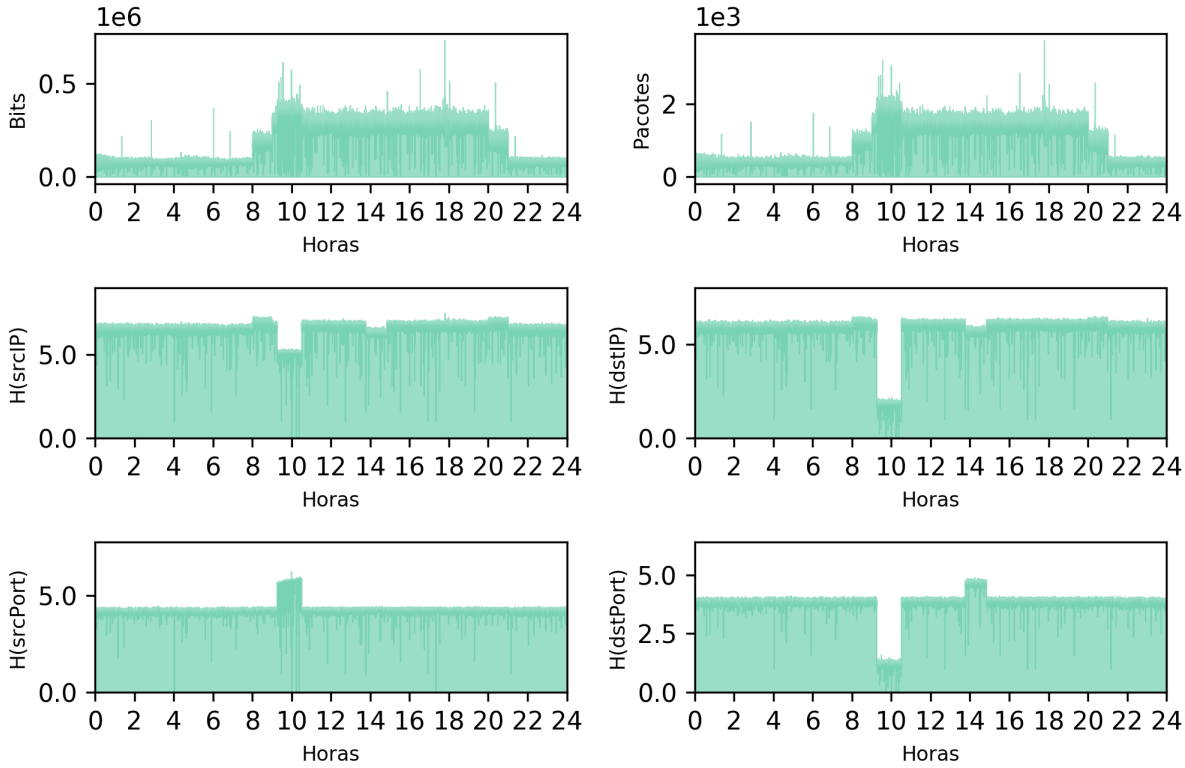


Figura 25 – Representação gráfica de cada um dos atributos coletados de um fluxo de rede, contendo um ataque DDoS das 09:15 às 10:30. Fonte: próprio autor.

Assim, a métrica é definida a partir do cálculo da frequência em que os números contidos no vetor dos valores previstos combinam com os números contidos no vetor dos valores esperados [91].

4.2.2 Erro Médio Quadrático Normalizado

O Erro Médio Quadrático Normalizado (*Normalized Mean Square Error*, NMSE) é outra métrica que se utiliza de dois vetores de tamanho n, de forma que um deles representa os valores observados de uma amostra e o outro representa os valores previstos sob essa mesma amostra. A partir do cálculo da métrica, quanto mais próximos os valores estiverem de zero, mais próximos entre si eles estarão.

Calculado a partir do somatório das diferenças quadradas entre duas variáveis, pelo produto da média de ambas. Equação 4.1 [92].

$$NMSE = \frac{\sum_{i=1}^{n} X_i - Y_i}{\sum_{i=1}^{n} Y_i} \tag{4.1}$$

O NMSE será utilizado para a avaliação da rede neural proposta neste trabalho, visto que essa métrica produziu bons resultados nos trabalhos [36][34], feitos pelo grupo de pesquisa de redes do Departamento de Computação da Universidade Estadual de Londrina.

4.2.3 Correlação de Pearson

O cálculo da correlação de Pearson produz valores contidos no intervalo $(-1, 1)$, que indicam quanto dois conjuntos de variáveis estão linearmente relacionados. O nome desta métrica é dada a como referência a Karl Pearson, que apresentou, em 1894, a equação para o cálculo de um coeficiente de correlação.

A equação 4.2 mostra como calcular a correlação de Pearson [93].

Uma correlação mais próxima de -1 indica que os conjuntos comparados se relacionam com proporção inversa; uma correlação resultante próxima de 0, indica que os conjuntos possuem pouca, ou nenhuma correlação; e um coeficiente próximo de 1 indica que os conjuntos possuem forte correlação linear de proporção direta [93].

$$r = \frac{n\sum(x_i \cdot y_i) - (\sum x_i)(\sum y_i)}{\sqrt{n\sum x_i^2 - (\sum x_i)^2} \cdot \sqrt{n\sum y_i^2 - (\sum y_i)^2}} \tag{4.2}$$

4.3 Pré-processamento

Para a implementação do modelo, foram utilizados seis dias de dados, em que cada dia consiste em um conjunto de dados de tráfego distribuídos ao longo de 86400 linhas, de forma que cada linha representa um segundo e o agrupamento total confere 24 horas de tráfego. As dimensões utilizadas foram as de *bytes*, entropia de IP de destino, entropia de porta de destino, entropia de IP de origem, entropia de porta de origem e pacotes. Todas essas dimensões estão organizadas como descrito acima, contendo 86400 linhas de informação de tráfego, de forma que cada linha representa um segundo; assim, cada uma destas dimensões equivale a seis conjuntos de arquivos de tráfego normal (sem anomalias), em que cada uma das dimensões foi reunida em colunas e cada dia de tráfego foi concatenado sequencialmente, formando uma matriz de dimensão 518400×6 como mostra a Tabela 4. Outro processo importante, realizado nesta matriz, foi a normalização dos valores para o intervalo de $[0, 1]$, de forma que pudessem ser introduzidos na rede neural para o processo de treinamento.

Depois, foi criada uma nova matriz a partir da matriz representada na Tabela 4, contendo apenas as dimensões nas quais as previsões foram realizadas. A representação do tempo na matriz original é dada para t, enquanto que na matriz dos dados de previsão é dada para $t+1$, com dimensão 518399×2, indicando os valores de saída esperados pela rede neural, de forma a caracterizar um problema de aprendizado supervisionado. Desta forma, será fornecido para o método os valores de t para que seja realizada a previsão de $t + 1$. A Tabela 5 compara lado a lado duas matrizes, sendo que a primeira representa a matriz original, mostrando apenas as dimensões utilizadas na previsão, distribuídos no tempo t, e a segunda representa a matriz dos valores esperados, utilizada para a previsão,

Tabela 4 – Matriz resultante da junção de 6 dias contendo todos os atributos.

	t	*Bytes*	H(dstIP)	H(dstPort)	H(srcIP)	H(srcPort)	Pacotes
Dia 1	0	1596	2.3219280	2.3219280	2.3219280	2.3219280	13
	1	19872	5.5359192	3.7685518	5.9925823	4.1748552	138
	2	19185	5.4290037	3.7923739	6.1292830	4.2663741	105
	3	39744	5.4778047	3.7972059	6.2479277	4.2950373	207
	4	61678	5.4257913	3.8184979	6.1898246	4.3073153	309
	5	82903	5.5280313	3.7748749	6.3037806	4.2809486	409
	6	80259	5.3491998	3.6581969	5.9541965	4.2695918	402
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	86398	68788	5.5240855	3.6400156	6.0194998	3.9253027	353
	86399	64816	5.5161934	3.7698705	5.9381156	4.0904260	347
Dia 2	86400	1644	1.5849625	0.9182959	1.5849625	1.5849625	7
	86401	17820	5.6753550	3.5297067	6.2288189	4.1205654	85
	86402	36852	5.6753550	3.5297067	6.2288189	4.1205654	181
	86403	59790	5.6753550	3.5297067	6.2288189	4.1205654	295
	86404	79942	5.5852900	3.5345736	6.2288189	4.1052957	401
	86405	58794	4.8329234	3.6189127	5.3575521	3.9395285	287
	86406	57577	5.0886297	3.3320954	5.3923173	3.7636597	281
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	172798	56276	5.5897245	3.8784049	6.1699252	3.9067380	303
	172799	62528	5.5248256	3.8170900	6.0660892	4.0795302	309
				⋮			
Dia 6	432000	13240	4.9267335	3.9193819	5.1699252	3.9511943	70
	432001	40696	6.3045626	3.8898263	6.8201790	4.2472992	213
	432002	85899	6.3045626	3.8898263	6.8201790	4.2472992	447
	432003	113388	6.3045626	3.8898263	6.8201790	4.2472992	589
	432004	127383	6.2199473	3.7953522	6.6582117	4.2647853	649
	432005	21312	6.1198044	3.9128323	6.5698557	4.1434069	111
	432006	41440	5.8224998	3.9819553	6.5541000	4.1009593	224
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	518398	83248	6.1945210	3.7838678	6.7681842	4.1372285	419
	518399	83700	5.9166532	3.7195318	6.4886384	4.2879815	432

distribuídos no tempo $t + 1$.

A partir da matriz contendo os atributos, em t, e a matriz dos valores esperados, em $t + 1$, foi gerada a matriz das amostras que foi dividida em um conjunto de testes e um conjunto de treino. Para a síntese dessa matriz, foi concatenada a matriz em $t + 1$ com a matriz em t, de forma a gerar uma matriz de dimensões 518399×8, como ilustra a Tabela 6.

O conjunto de testes, ou *testing set*, foi obtido a partir de um dia inteiro de tráfego normal (dia 6), retirado do agrupamento, derivando uma matriz de dimensões 86400×6, que corresponde a aproximadamente $16,667\%$ do total. O conjunto de treino, ou *training set*, foi obtido a partir do restante do grupo (dias 1 a 5), resultando numa matriz de

Tabela 5 – Comparação das matrizes de valores de entrada e de valores esperados, em que a primeira encontra-se no tempo t e a segunda, no tempo $t + 1$.

t		H(dstIP)	H(dstPort)	
0		2.3219280	2.3219280	
1		5.5359192	3.7685518	
2		5.4290037	3.7923739	
3		5.4778047	3.7972059	
4		5.4257913	3.8184979	
5		5.5280313	3.7748749	
6		5.3491998	3.6581969	
⋮		⋮	⋮	
86398		5.5240855	3.6400156	
86399	...	5.5161934	3.7698705	...
⋮		⋮	⋮	
432000		4.9267335	3.9193819	
432001		6.3045626	3.8898263	
432002		6.3045626	3.8898263	
432003		6.3045626	3.8898263	
432004		6.2199473	3.7953522	
432005		6.1198044	3.9128323	
432006		5.8224998	3.9819553	
⋮		⋮	⋮	
518398		6.1945210	3.7838678	
518399		5.9166532	3.7195318	

(a) Matriz dos valores de entrada.

$t + 1$	H(dstIP)	H(dstPort)
0	5.5359192	3.7685518
1	5.4290037	3.7923739
2	5.4778047	3.7972059
3	5.4257913	3.8184979
4	5.5280313	3.7748749
5	5.3491998	3.6581969
⋮	⋮	⋮
86397	5.5240855	3.6400156
86398	5.5161934	3.7698705
⋮	⋮	⋮
431999	4.9267335	3.9193819
432000	6.3045626	3.8898263
432001	6.3045626	3.8898263
432002	6.3045626	3.8898263
432003	6.2199473	3.7953522
432004	6.1198044	3.9128323
432005	5.8224998	3.9819553
⋮	⋮	⋮
518397	6.1945210	3.7838678
518398	5.9166532	3.7195318

(b) Matriz dos valores esperados.

dimensões 431999×6, totalizando 431999 amostras de treinamento.

4.4 Rede neural

4.4.1 LSTM

A rede neural definida pelo modelo possui uma camada de entrada com seis valores, representando as seis dimensões coletadas do tráfego, assim como é mostrado nos valores de entrada na Tabela 6. A segunda camada da rede neural ou, neste caso, a primeira camada oculta, é uma rede LSTM que contém 6 neurônios; é caracterizada como uma rede recorrente do tipo um-para-um e realiza uma conexão com uma camada densa, que por sua vez produz os dados de saída, possuindo tamanho 2, representando as duas dimensões em que foram feitas a previsão. Uma representação visual da rede neural implementada pode ser vista na figura 26.

4.4.2 GRU

A definição da rede neural GRU foi semelhante à LSTM, de forma que a rede neural em questão também possui uma camada de entrada com seis valores, que representam

Tabela 6 – Representação da matriz que contém os valores de entrada e os valores esperados da rede neural.

	Valores de entrada					**Valores esperados**	
t		H(dstIP)	H(dstPort)		Pacotes	H(dstIP)	H(dstPort)
0		0.2174457	0.5612527		0.0030552	0.7461193	0.9109283
1		0.7461193	0.9109283		0.0324324	0.7285326	0.9166865
2		0.7285326	0.9166865		0.0246769	0.7365600	0.9178545
3		0.7365600	0.9178545		0.0486486	0.7280042	0.9230012
4		0.7280042	0.9230012		0.0726204	0.7448218	0.9124567
5		0.7448218	0.9124567		0.0961222	0.7154056	0.8842535
6		0.7154056	0.8842535		0.0944771	0.7189693	0.9334570
⋮		⋮	⋮		⋮	⋮	⋮
86397		0.7377681	0.9152752		0.0864865	0.7441728	0.8798587
86398		0.7441728	0.8798587		0.0829612	0.7428746	0.9112471
⋮	⋯	⋮	⋮	⋯	⋮	⋮	⋮
432001		0.8725544	0.9402426		0.0500588	0.8725544	0.9402426
432002		0.8725544	0.9402426		0.1050529	0.8725544	0.9402426
432003		0.8725544	0.9402426		0.1384254	0.8586360	0.9174064
432004		0.8586360	0.9174064		0.1525264	0.8421633	0.9458035
432005		0.8421633	0.9458035		0.0260870	0.7932593	0.9625118
432006		0.7932593	0.9625118		0.0526439	0.8725544	0.9402426
432007		0.8725544	0.9402426		0.0484136	0.8725544	0.9402426
⋮		⋮	⋮		⋮	⋮	⋮
518397		0.8201721	0.9226764		0.1008226	0.8544536	0.9146305
518398		0.8544536	0.9146305		0.0984724	0.8087468	0.8990793

as seis dimensões coletadas do tráfego. A primeira camada oculta da rede neural é uma rede GRU contendo 5 neurônios, caracterizada como uma rede recorrente do tipo um-para-um, realizando uma conexão densa com a camada de saída, que possui tamanho 2, representando as duas dimensões em que foi realizada a previsão. Esta rede neural pode ser visualizada na figura 27.

Para a decisão dos parâmetros utilizados por ambas as redes neurais, foi realizada uma séries de testes com o intuito de definir quais teriam o maior potencial de produzir bons resultados na etapa de detecção da anomalia. Este processo é descrito na sessão 5.1.

4.4.3 Valores de saída

Os valores de saída produzidos pela rede neural são equivalentes às dimensões de entropia de IP de destino e entropia de porta de destino, representando um dia todo de tráfego. Sendo assim, a saída é uma matriz de dimensões 86400×2, contendo valores no intervalo de $[0, 1]$ que são utilizados para o cálculo das métricas. Nessa mesma matriz é aplicada a transformada inversa da normalização, fazendo com que os valores retornem para seus intervalos originais, como mostram as Tabelas 7 e 8.

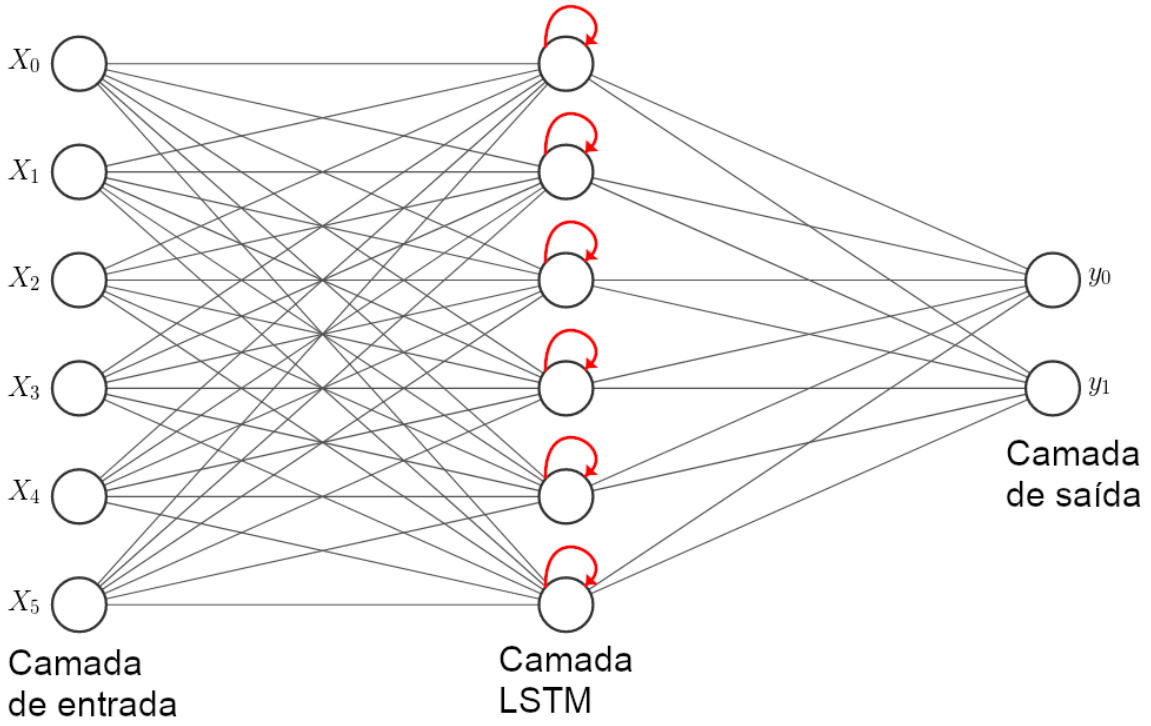


Figura 26 – Configuração da rede neural LSTM implementada. Fonte: próprio autor.

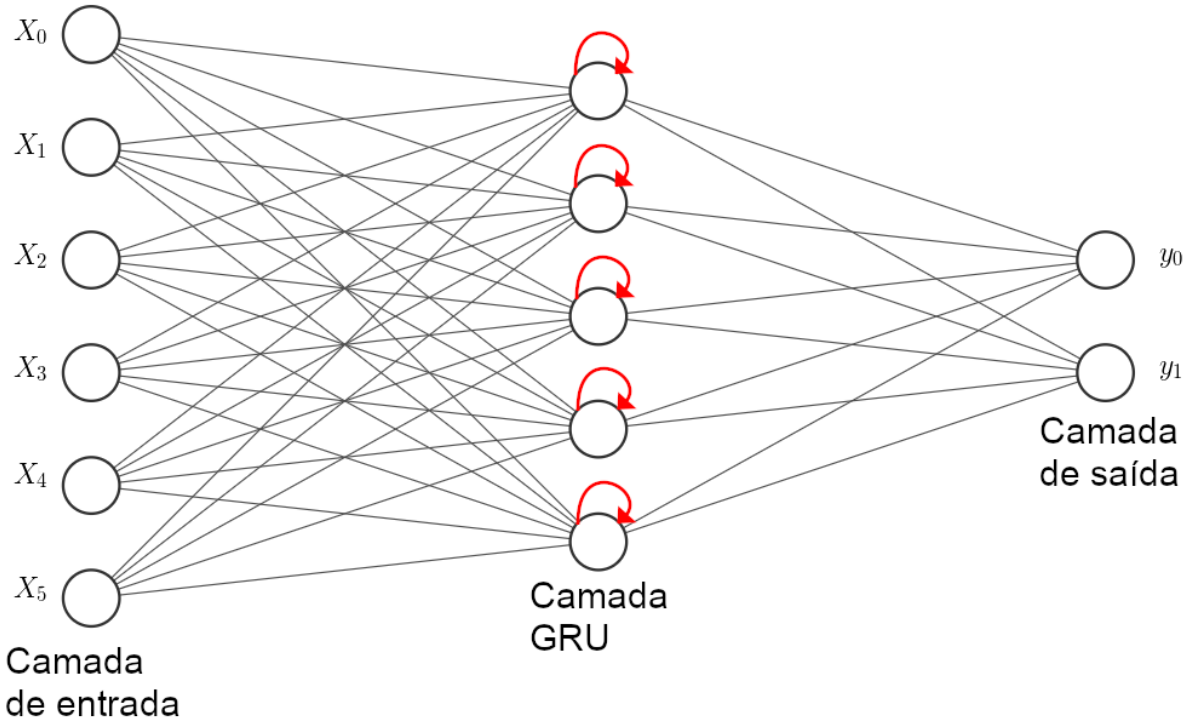


Figura 27 – Configuração da rede neural GRU implementada. Fonte: próprio autor.

Tabela 7 – Valores de saída produzidos pela rede neural LSTM.

t	H(dstIP)	H(dstPort)
0	6.5522275	3.9324186
1	5.9695286	4.0588611
2	6.4644689	3.9242275
3	6.4740707	3.9205772
4	6.4798959	3.9183508
5	6.4591169	3.8584790
6	6.4212882	3.9570255
⋮	⋮	⋮
86398	6.4094453	3.8800036
86399	6.4482585	3.8568988

Tabela 8 – Valores de saída produzidos pela rede neural GRU.

t	H(dstIP)	H(dstPort)
0	6.605469	3.92306
1	5.851216	3.998203
2	6.452236	3.917406
3	6.454591	3.911044
4	6.456719	3.906516
5	6.424969	3.836136
6	6.37008	3.947079
⋮	⋮	⋮
86398	6.303043	3.866842
86399	6.397285	3.836968

5 RESULTADOS OBTIDOS - LSTM

Nesta sessão, serão levantadas discussões a respeito dos métodos utilizados para avaliar o tráfego gerado pela rede neural LSTM, bem como as metodologias escolhidas para realizar a detecção da anomalia de DDoS. Depois, será discorrido sobre os resultados obtidos no treinamento da rede neural, na qualidade de previsão do tráfego gerado pelo método e as conclusões obtidas a partir de um estudo de caso em que foi realizada a detecção da anomalia de DDoS.

5.1 Treinamento e validação

Para o treinamento da rede neural foi necessário decidir alguns parâmetros que têm impacto direto com os resultados gerados pelo método. Os parâmetros em questão são a quantidade de neurônios da rede neural e o valor de épocas, em que uma época equivale a uma iteração completa entre o conjunto de treinamento, de forma que n épocas representam n iterações pelo conjunto, ou seja, a quantidade total de iterações da rede neural é dada pelo produto da quantidade de amostras pela quantidade de épocas.

Com a finalidade de decidir qual seria a melhor combinação de valores de épocas e quantidade de neurônios, a rede neural foi treinada para seis valores diferentes de épocas, dados por $epocas = 5, 10, 15, 20, 25, 30$; sendo que, para cada valor de época, o modelo foi testado para 1 a 10 neurônios na camada LSTM. Este processo totalizou 60 treinamentos diferentes realizados na rede neural. Os resultados são observados na Tabela 9, em que foram feitas comparações utilizando a métrica de acurácia para cada combinação de treinamento.

Tabela 9 – Acurácia obtida em função de Neurônios × Épocas

	Épocas					
Neurônios	5	10	15	20	25	30
1	**97.71%**	97.77%	97.77%	97.73%	97.72%	97.74%
2	97.75%	97.77%	97.84%	98.12%	97.94%	98.09%
3	97.76%	98.05%	98.05%	98.06%	98.08%	97.99%
4	97.76%	97.74%	98.01%	98.01%	97.95%	98.01%
5	97.77%	97.93%	98.00%	98.01%	98.07%	98.00%
6	97.76%	97.90%	97.96%	97.91%	98.02%	**98.15%**
7	97.77%	97.84%	97.99%	98.03%	98.02%	97.94%
8	97.76%	97.80%	98.06%	97.90%	97.98%	97.96%
9	97.76%	97.84%	97.97%	98.04%	97.97%	97.97%
10	97.78%	97.78%	97.96%	97.99%	97.95%	97.93%

Com isso foi possível concluir que o melhor resultado foi obtido para 6 neurônios com 30 épocas. Desta forma, foi possível avançar para a próxima etapa, em que foram testados métodos para a detecção da anomalia de DDoS.

5.1.1 Detecção de anomalias

Na etapa de detecção de anomalias foram feitas análises num tráfego a fim de indicar atividades que desviem de um comportamento normal. Para isso foi analisada a divergência entre um tráfego sem anomalias, gerado pela LSTM, com uma assinatura de tráfego contendo um ataque DDoS. A Figura 28 mostra uma comparação entre três tipos de tráfego nas duas dimensões analisadas, representadas pelas colunas da imagem, sendo que cada um dos gráficos simboliza 24 horas úteis de dados. Na primeira linha da figura consta um tráfego real, sem ataque, gerado pelo *Mininet*, utilizado como conjunto de testes da rede neural; na segunda linha, o tráfego gerado pelo LSTM, sem ataque; e na terceira linha, um tráfego contendo uma anomalia de DDoS das 9:15 às 10:30 (intervalo destacado em vermelho).

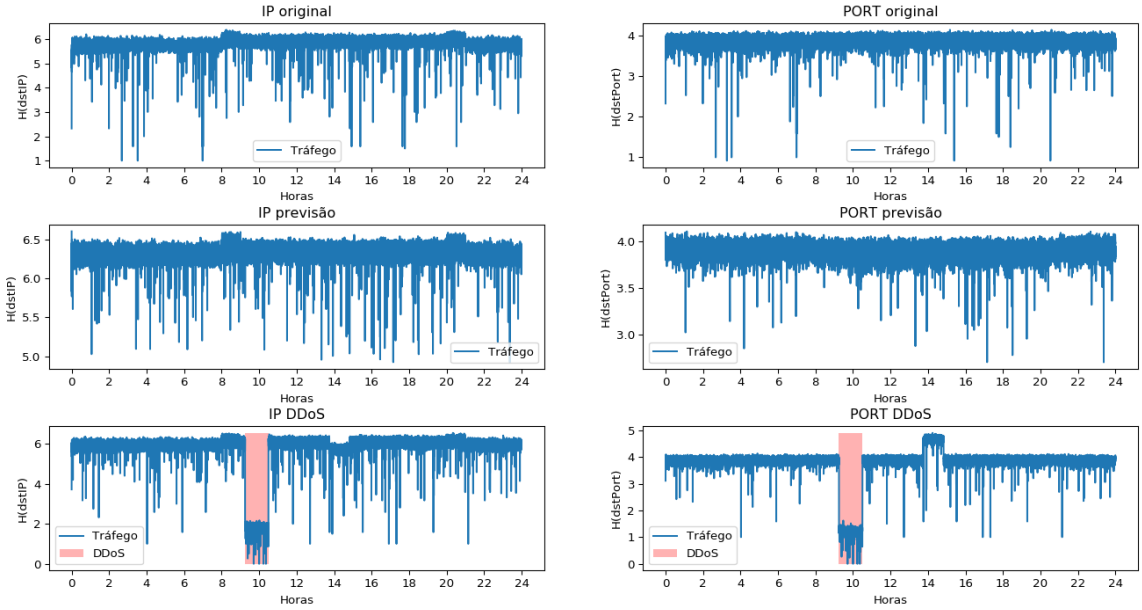


Figura 28 – Na primeira linha, os parâmetros originais, sem ataque; na segunda, os parâmetros gerados pela rede neural e na terceira linha, os parâmetros com DDoS. Fonte: próprio autor.

Para examinar a existência de uma anomalia de DDoS no tráfego analisado, foi aplicado um limiar superior e um limiar inferior. Em seguida, os valores foram aplicados no tráfego gerado pela rede neural, gerando intervalos de confiança que foram, então, comparados com um cenário anômalo.

O processo para a definição dos limiares foi realizada a partir da execução de três passos:

- Passo 1: O tráfego gerado pela rede LSTM foi multiplicado por uma taxa de *threshold* em porcentagem, indicando o intervalo de confiança que foi utilizado na detecção da anomalia. Este processo gerou dois conjuntos de dados indicando, respectivamente, o tráfego gerado acrescido do limiar e o tráfego gerado reduzido do limiar. Este resultado pode ser visualizado para a dimensão *H(dstIP)* na ilustração 29 (o processo é análogo para a dimensão *H(dstPort)*).

- Passo 2: Após a obtenção dos dois conjuntos de limiares, cada um deles foi divido em 60 fragmentos iguais e foi calculada a média de cada uma das partes geradas, para cada conjunto, gerando, então, o intervalo de confiança final (figura 30), que foi utilizado para a detecção do DDoS.

- Passo 3: Por fim, foi realizada a concatenação lado a lado da matriz resultante da rede neural com a matriz das dimensões contendo DDoS; a matriz resultante deste processo foi normalizada com valores entre 0 e 1 e depois separada novamente em matriz resultante e matriz de DDoS (tabela 10). O intuito deste procedimento é normalizar os intervalos de confiança com relação ao tráfego anômalo para o procedimento de detecção da anomalia.

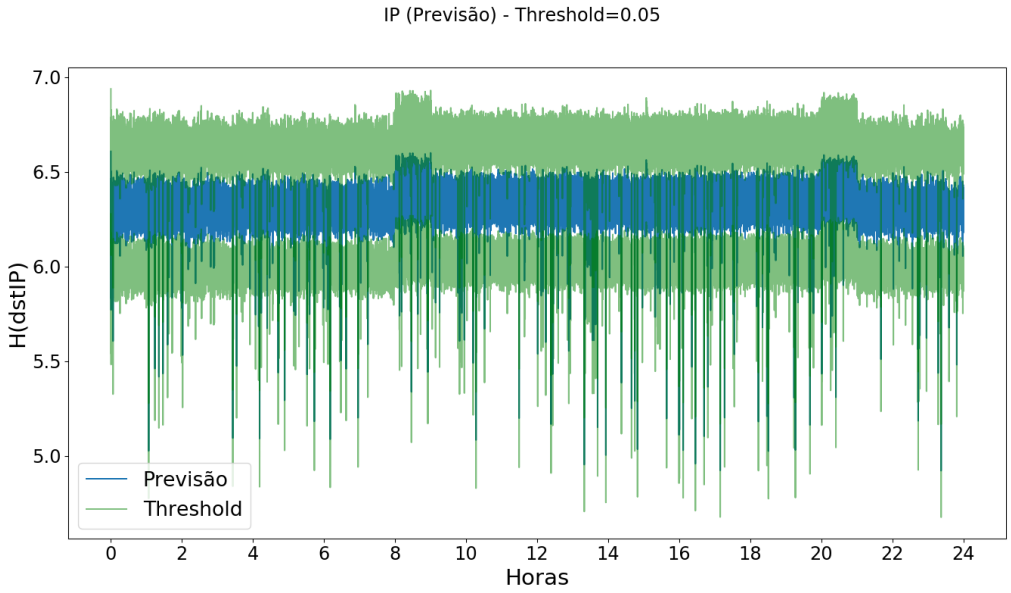


Figura 29 – Aplicação dos limiares inferior e superior no tráfego gerado pela rede LSTM, gerando os dois tráfegos exibidos em verde. Fonte: próprio autor.

Deste modo, para detectar a anomalia pegou-se os dois conjuntos gerados pelo processo descrito anteriormente e a comparação com o tráfego anômalo se deu de maneira que todo o tráfego que estivesse contido dentro do intervalo de confiança era considerado normal e todo o tráfego fora dele era considerado anômalo. Para encontrar os melhores limiares, foi feita a aplicação de taxas de *threshold* de $5\%, 10\%$, 15%, 20% e 25%. Estes limiares podem ser visualizados nas Figuras 31, 32, 33, 34 e 35.

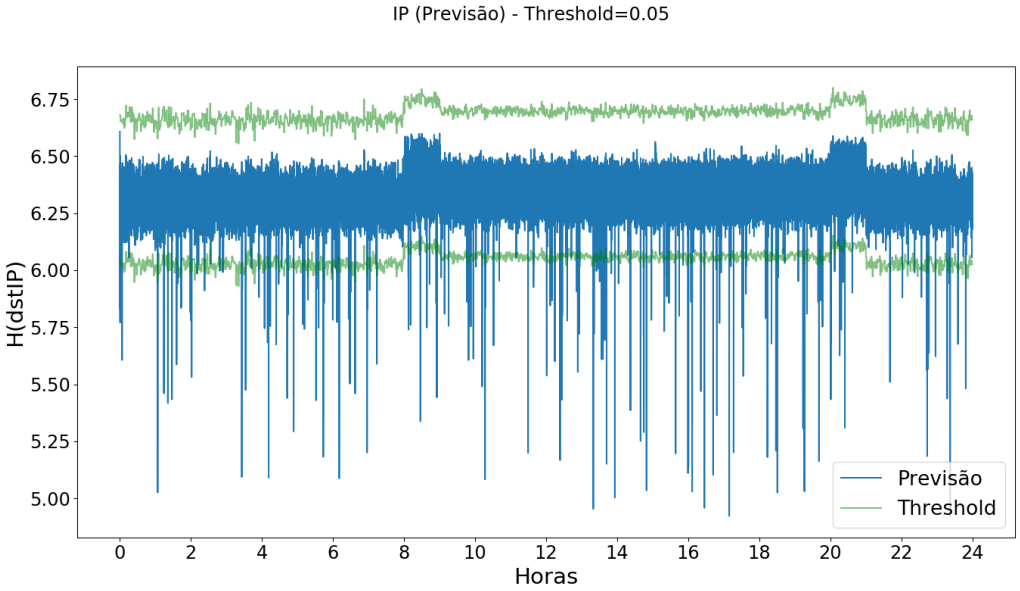


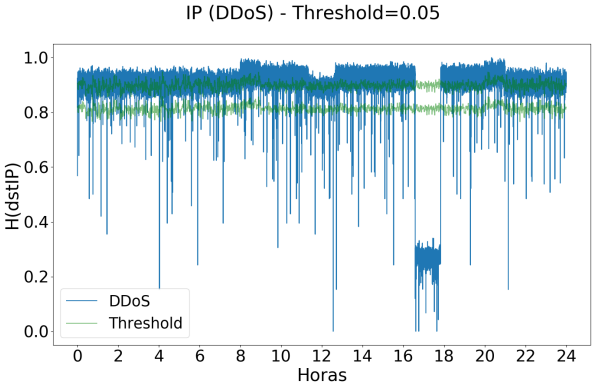
Figura 30 – Geração do intervalo de confiança a partir da média do resultante dos limiares. Fonte: próprio autor.

Tabela 10 – Matriz normalizada resultante da concatenação das dimensões geradas pela LSTM com as dimensões contendo um ataque DDoS.

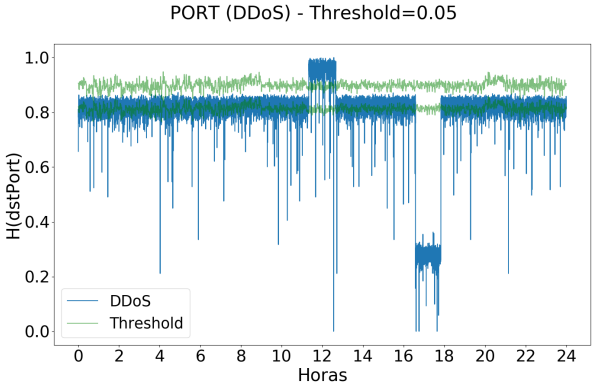
	Dimensões Geradas		**Dimensões Anômalas**	
t	H(dstIP)	H(dstPort)	H(dstIP)	H(dstPort)
0	1.0	0.8780672	0.5684728	0.6349355
1	0.5894034	0.9920795	0.9190009	0.7776738
2	0.9137682	0.8945412	0.9190009	0.7776738
3	0.9016532	0.8676869	0.9221142	0.7743944
4	0.8980309	0.8586381	0.9191768	0.77568
5	0.8725792	0.8042784	0.7876606	0.7067228
6	0.8781431	0.9352381	0.8408086	0.7700408
⋮	⋮	⋮	⋮	⋮
86398	0.814165	0.8291982	0.9125735	0.8030158
86399	0.8728324	0.811507	0.9334215	0.8154143

Os resultados das métricas foram calculados a partir dos valores de Verdadeiros Positivos (VP), indicando pontos anômalos que foram detectados como tal; Verdadeiros Negativos (VN), para pontos não anômalos detectados como tal; Falsos Positivos (FP), sendo pontos não anômalos identificados como anômalos; e Falsos Negativos (FN), que são pontos anômalos identificados como não anômalos. Com base nestes resultados, foram calculadas as métricas de Precisão, Revocação, Acurácia e Taxa de Falsos Positivos, cujas fórmulas são respectivamente descritas pelas equações 5.1, 5.2, 5.3 e 5.4.

$$precisão = \frac{VP}{VP + FP} \tag{5.1}$$

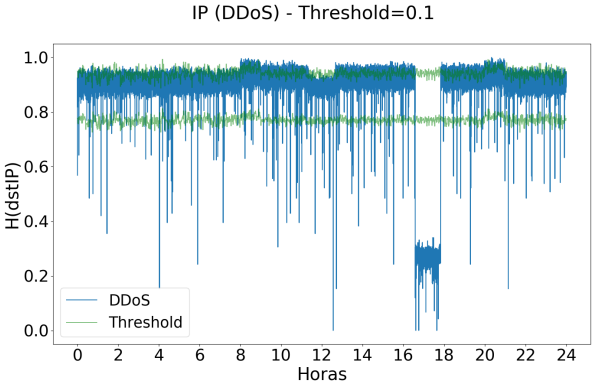


(a) Entropia de IP de destino.

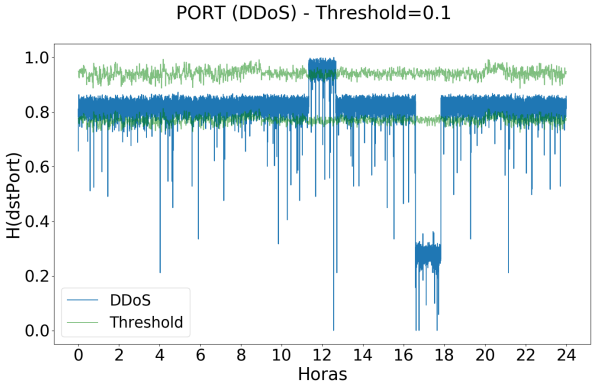


(b) Entropia de portas destino.

Figura 31 – Limiares de 5%, aplicados às dimensões analisadas. Fonte: próprio autor.



(a) Entropia de IP de destino.



(b) Entropia de portas destino.

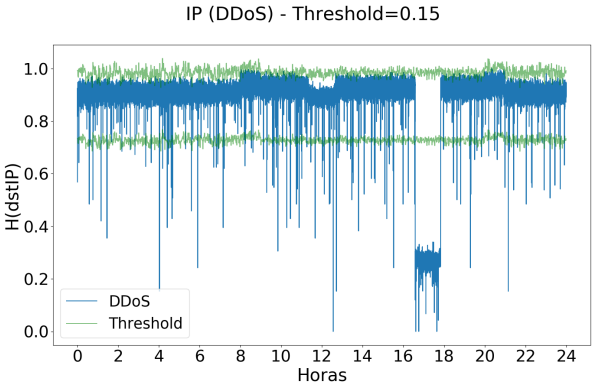
Figura 32 – Limiares de 10%, aplicados às dimensões analisadas. Fonte: próprio autor.

$$revocação = \frac{VP}{VP + FN} \tag{5.2}$$

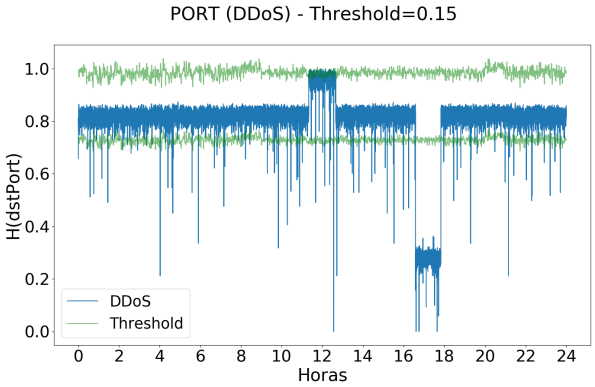
$$acurácia = \frac{VP + VN}{VP + VN + FP + FN} \tag{5.3}$$

$$taxa\ de\ f.p. = \frac{FP}{FP + TN} \tag{5.4}$$

O resultado das métricas para a entropia de IP de destino é ilustrado pela tabela 11 e para a entropia de portas de destino, pela tabela 12.

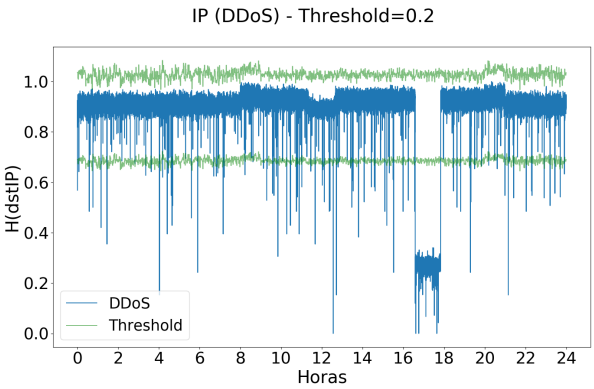


(a) Entropia de IP de destino.

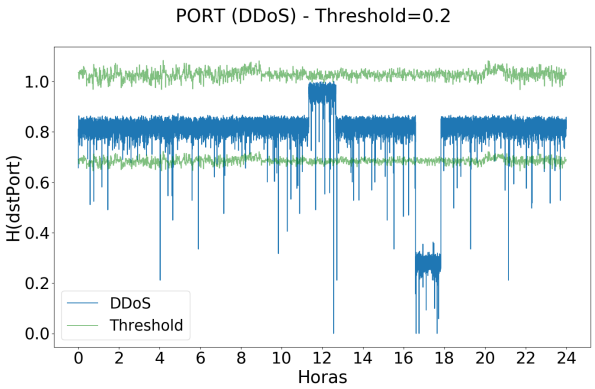


(b) Entropia de portas destino.

Figura 33 – Limiares de 15%, aplicados às dimensões analisadas. Fonte: próprio autor.

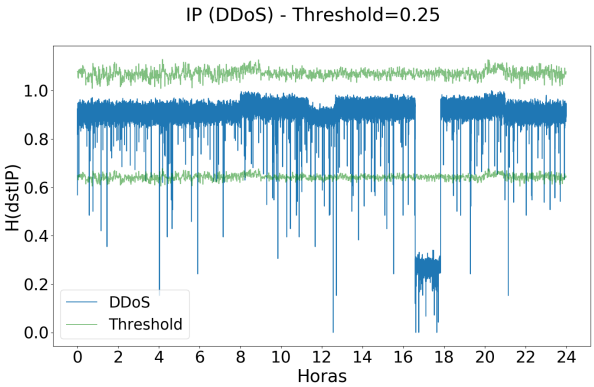


(a) Entropia de IP de destino.

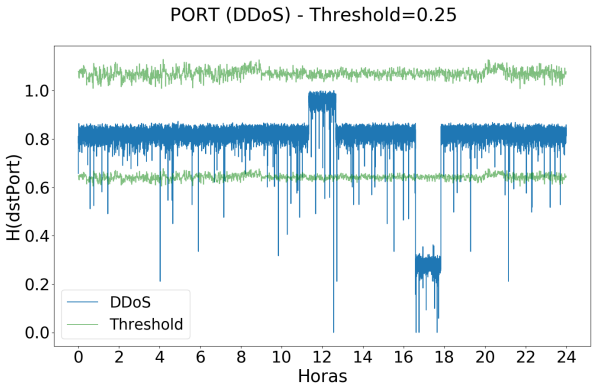


(b) Entropia de portas destino.

Figura 34 – Limiares de 20%, aplicados às dimensões analisadas. Fonte: próprio autor.

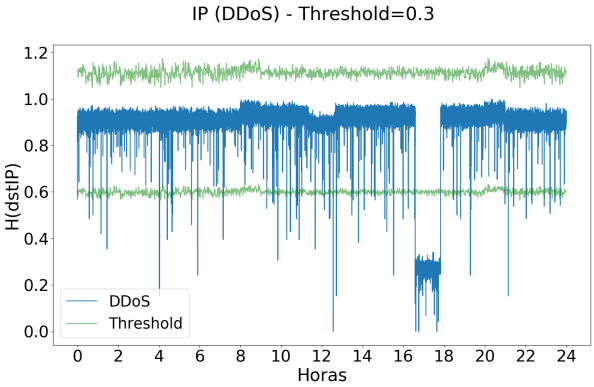


(a) Entropia de IP de destino.

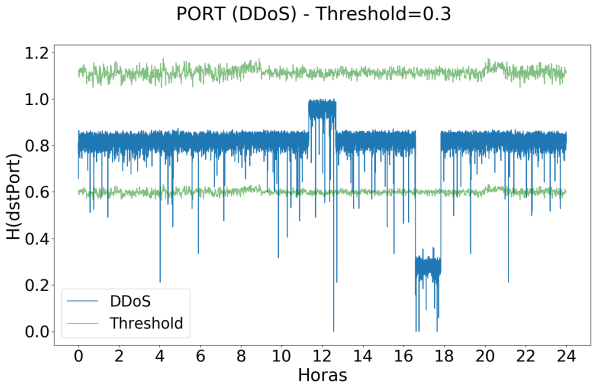


(b) Entropia de portas destino.

Figura 35 – Limiares de 25%, aplicados às dimensões analisadas. Fonte: próprio autor.

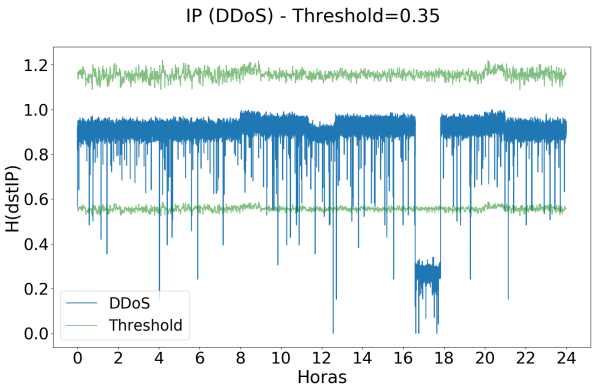


(a) Entropia de IP de destino.

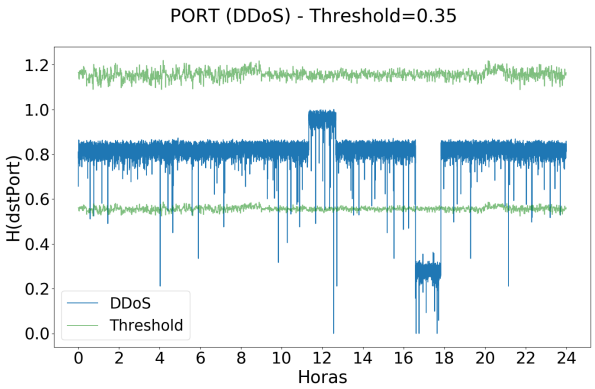


(b) Entropia de portas destino.

Figura 36 – Limiares de 30%, aplicados às dimensões analisadas. Fonte: próprio autor.

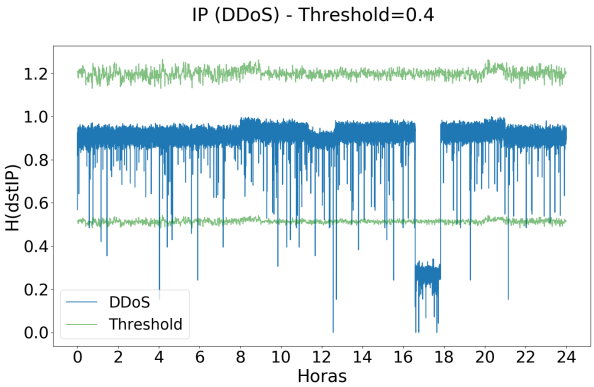


(a) Entropia de IP de destino.

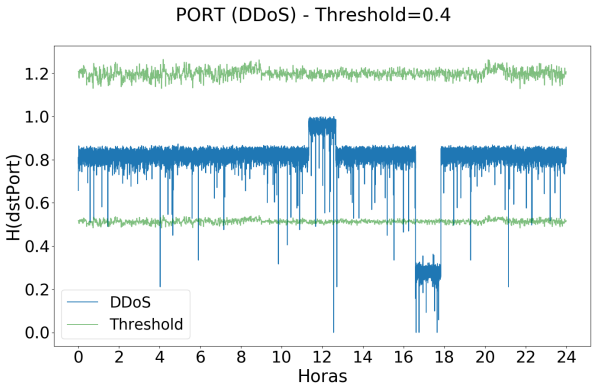


(b) Entropia de portas destino.

Figura 37 – Limiares de 35%, aplicados às dimensões analisadas. Fonte: próprio autor.

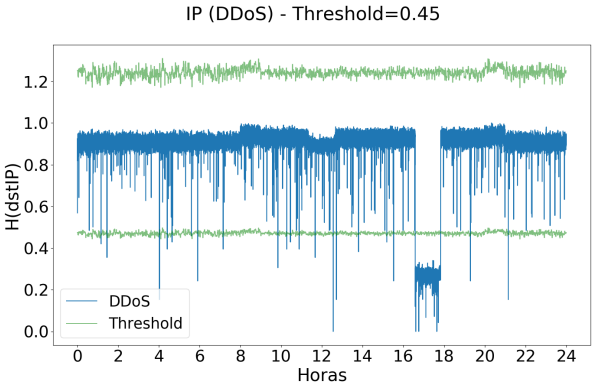


(a) Entropia de IP de destino.

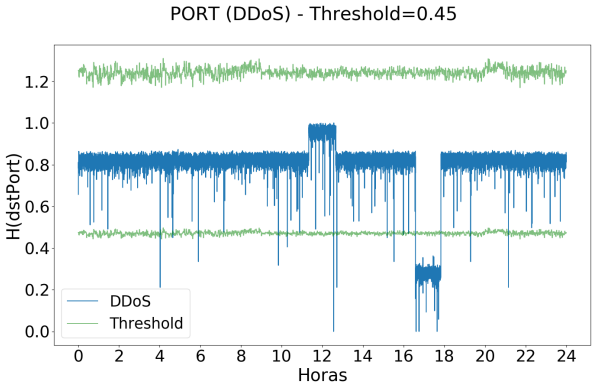


(b) Entropia de portas destino.

Figura 38 – Limiares de 40%, aplicados às dimensões analisadas. Fonte: próprio autor.

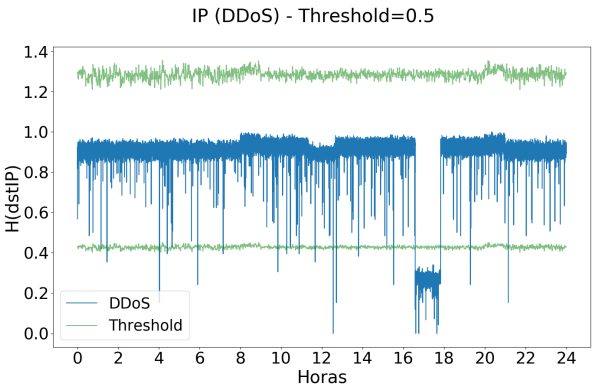


(a) Entropia de IP de destino.

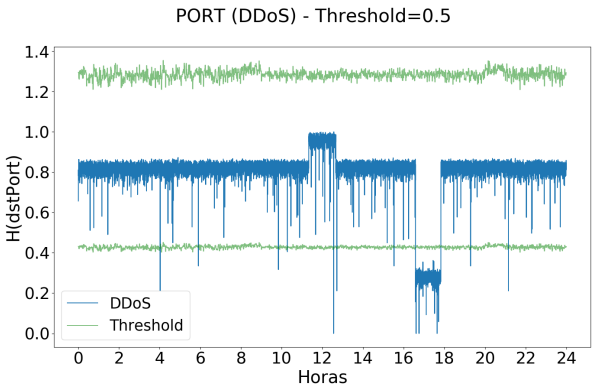


(b) Entropia de portas destino.

Figura 39 – Limiares de 45%, aplicados às dimensões analisadas. Fonte: próprio autor.

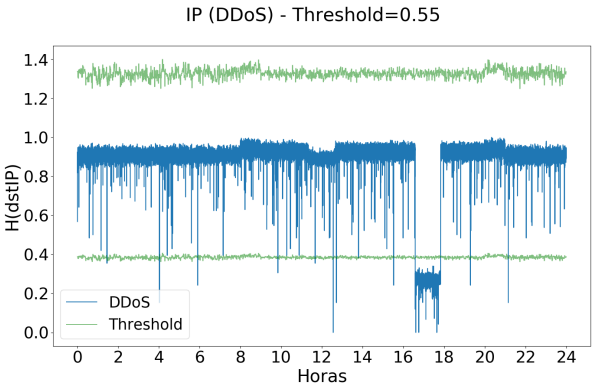


(a) Entropia de IP de destino.

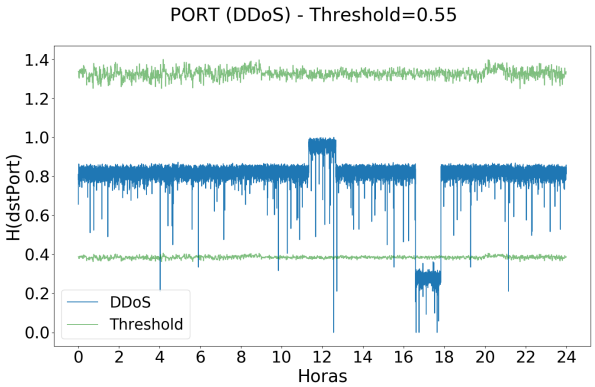


(b) Entropia de portas destino.

Figura 40 – Limiares de 50%, aplicados às dimensões analisadas. Fonte: próprio autor.

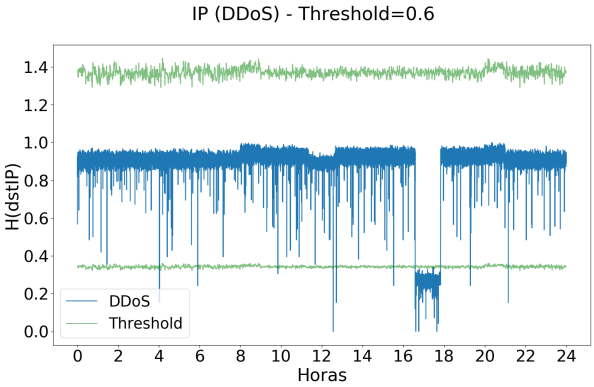


(a) Entropia de IP de destino.

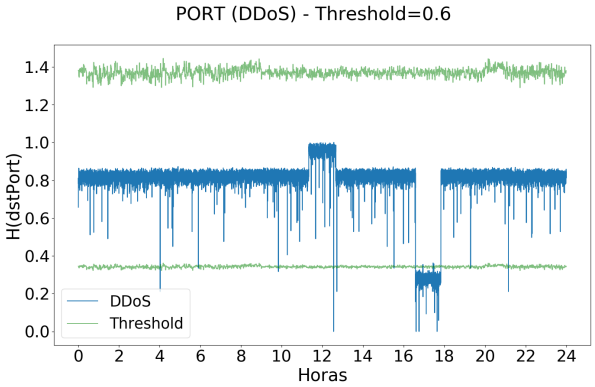


(b) Entropia de portas destino.

Figura 41 – Limiares de 55%, aplicados às dimensões analisadas. Fonte: próprio autor.

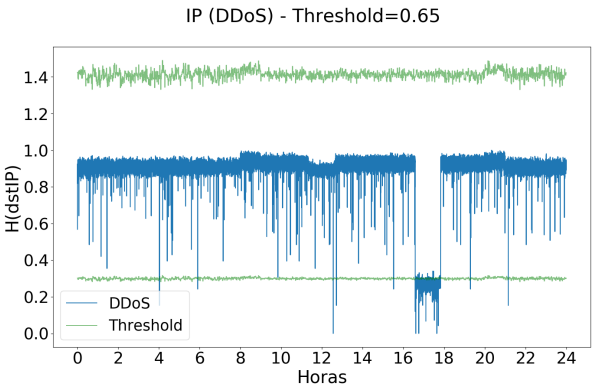


(a) Entropia de IP de destino.

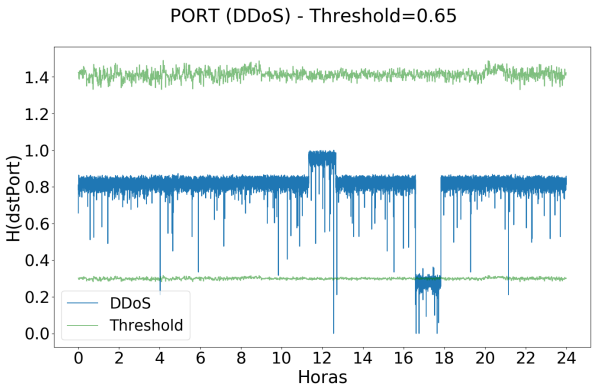


(b) Entropia de portas destino.

Figura 42 – Limiares de 60%, aplicados às dimensões analisadas. Fonte: próprio autor.

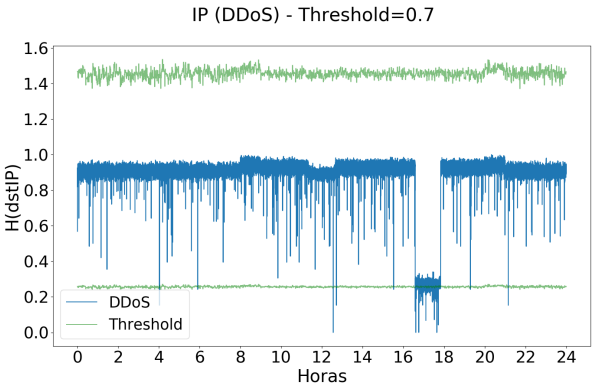


(a) Entropia de IP de destino.

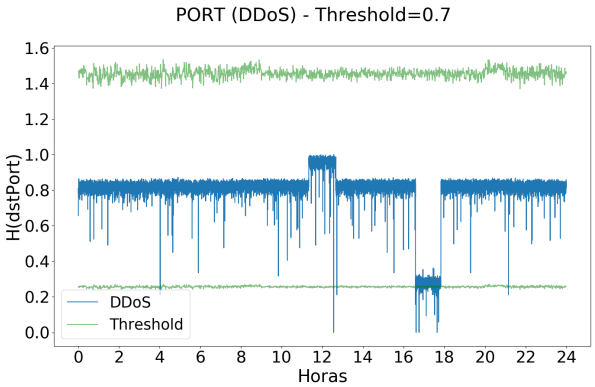


(b) Entropia de portas destino.

Figura 43 – Limiares de 65%, aplicados às dimensões analisadas. Fonte: próprio autor.

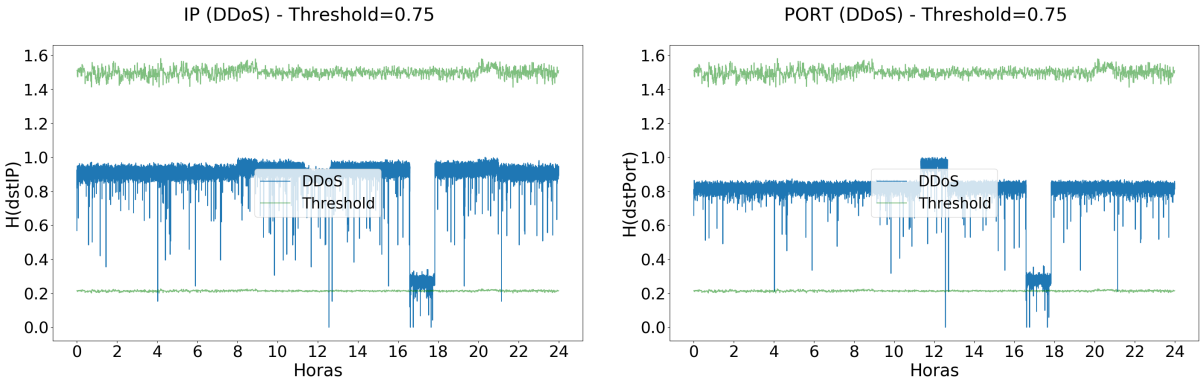


(a) Entropia de IP de destino.



(b) Entropia de portas destino.

Figura 44 – Limiares de 70%, aplicados às dimensões analisadas. Fonte: próprio autor.



(a) Entropia de IP de destino.

(b) Entropia de portas destino.

Figura 45 – Limiares de 75%, aplicados às dimensões analisadas. Fonte: próprio autor.

Tabela 11 – Resultado das métricas no conjunto de entropias de IP de destino.

	Métricas			
Limiares	Precisão	Revocação	Acurácia	Taxa de f.p.
5%	6.07%	100%	19.44%	84.99%
10%	17.12%	100%	74.78%	26.60%
15%	81.45%	100%	98.81%	1.25%
20%	97.89%	100%	99.89%	0.12%
25%	98.51%	100%	99.92%	0.08%
30%	98.79%	100%	99.94%	0.07%
35%	99.01%	100%	99.95%	0.05%
40%	99.23%	100%	99.96%	0.04%
45%	99.51%	100%	99.97%	0.03%
50%	99.58%	100%	99.98%	0.02%
55%	99.71%	100%	99.98%	0.02%
60%	99.76%	100%	99.99%	0.01%
65%	99.77%	94.42%	99.70%	0.01%
70%	96.61%	6.33%	95.11%	0.01%
75%	88.64%	0.87%	94.83%	0.01%

Tabela 12 – Resultado das métricas no conjunto de entropias de portas de destino.

	Métricas			
Limiares	Precisão	Revocação	Acurácia	Taxa de f.p.
5%	6.85%	100%	29.20%	74.69%
10%	17.50%	100%	75.45%	25.90%
15%	65.62%	100%	97.27%	2.88%
20%	97.02%	100%	99.84%	0.17%
25%	98.71%	100%	99.93%	0.07%
30%	98.99%	100%	99.95%	0.06%
35%	99.25%	100%	99.96%	0.04%
40%	99.40%	100%	99.97%	0.03%
45%	99.58%	100%	99.98%	0.02%
50%	99.73%	100%	99.99%	0.01%
55%	99.78%	100%	99.99%	0.01%
60%	99.80%	100%	99.99%	0.01%
65%	99.89%	99.89%	99.99%	0.01%
70%	99.82%	63.31%	98.08%	0.01%
75%	91.67%	1.22%	94.85%	0.01%

6 ANÁLISE DOS RESULTADOS - LSTM

Neste capítulo serão analisados os resultados apresentados no Capítulo 5, com o intuito de apresentar qual foi o melhor e o pior cenário de treinamento da rede neural, e quais foram os melhores limiares utilizados na detecção de anomalias. Além disso, serão apresentadas tabelas com a validação da predição das dimensões de entropia de IP de destino e entropia de portas de destino utilizando as métricas de NMSE e a Correlação de Pearson, discutidas anteriormente na seção 4.2.

6.1 Cenários de treinamento

A partir da análise da Tabela 9, descrita na seção 5.1, que mostra as comparações entre os treinamentos utilizando a métrica de acurácia para cada uma das combinações de neurônios e épocas, é possível observar que a maior soma de acurácias foi obtida para o valor de 3 neurônios. Entretanto, a combinação de treinamento (Neurônios × Épocas) que obteve a melhor acurácia em relação a todos os outros casos, foi a configuração na qual são caracterizados 6 neurônios e 30 épocas. Na Figura 46 é possível observar o valor da acurácia em função durante o treinamento deste caso.

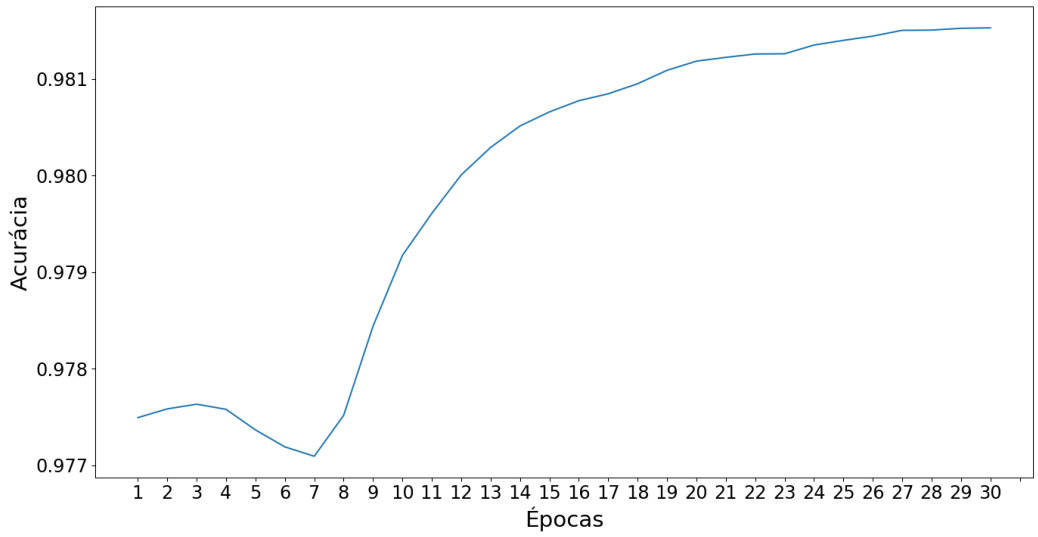


Figura 46 – Evolução da acurácia para 6 neurônios, em função do número de épocas. Fonte: próprio autor.

Ainda analisando a Tabela 9, é possível observar que o pior caso foi a combinação de 1 neurônio com 5 épocas. A evolução da acurácia para este cenário pode ser observada na figura 47.

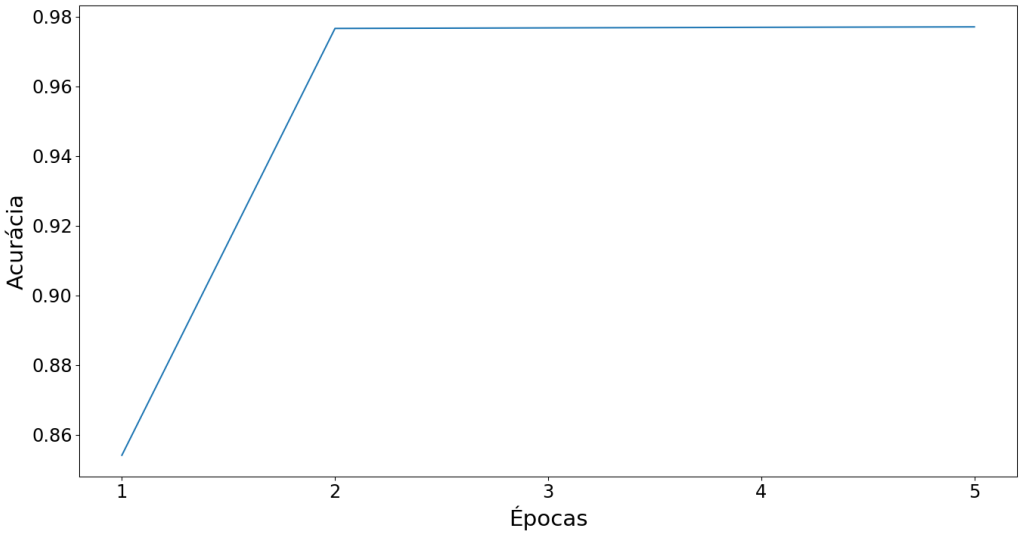


Figura 47 – Evolução da acurácia para 1 neurônio, em função do número de épocas. Fonte: próprio autor.

A partir dos resultados produzidos pela melhor combinação, foram geradas as figuras 26 e 28, em que a primeira mostra a rede neural implementada e a segunda mostra a comparação entre o tráfego gerado, com um tráfego real não anômalo e um tráfego real contendo uma anomalia de DDoS. Da mesma forma, foram geradas as seguintes tabelas: Tabela 10, quando apresenta os valores resultantes para as dimensões geradas; tabelas 11 e 12, que ilustram, respectivamente, os resultados das métricas calculadas para as dimensões de entropia de IP de destino e entropia de portas de destino.

6.2 Melhores limiares na detecção de anomalias

Como é possível observar nas tabelas 11 e 12, em todos os limiares de 5% até 60% foi possível detectar todo o intervalo anômalo sem a existência de nenhum falso negativo (a partir da análise das revocações). Todavia, a taxa de falsos positivos é alta para os limiares de $5\%, 10\%$ e 15%, como constatado nos resultados obtidos para as acurácias correspondentes. Além disso, ao observar os resultados dos limiares estritamente maiores do que 60% é possível notar que a revocação começa a diminuir, isto é consequência do limiar ter se tornado grande o bastante para englobar o tráfego anômalo e considerá-lo como tráfego normal, como é possível observar nas figuras 43, 44 e 45.

Os melhores resultados foram para limiares entre 20% e 60%. As figuras 34, 35, 36, 37, 38, 39, 40, 41 e 42 mostram o esboço dos limiares superior e inferior em relação ao tráfego para ambas as dimensões analisadas, respectivamente, indicando, a partir dos *thresholds*, uma boa separação entre tráfego anômalo e tráfego normal. Entretanto, é possível notar que limiares muito grandes podem incorporar o tráfego anômalo, fazendo

com que o mesmo seja considerado tráfego normal durante o processo de análise.

6.3 Validação da previsão

Para validar as previsões realizadas pela rede neural, as métricas propostas são as de NMSE e a correlação de Pearson. Para computar os valores, foram consideradas as duas dimensões estudadas e o conjunto de testes, de forma que ambos estavam normalizados no intervalo entre 0 e 1, como mostra a tabela 13; o cálculo foi realizado para cada uma das dimensões.

Tabela 13 – Matrizes dos conjuntos de predição e conjunto de testes.

t	H(dstIP)	H(dstPort)
0	1.0	0.8780672
1	0.5894034	0.9920795
2	0.9137682	0.8945412
3	0.9016532	0.8676869
4	0.8980309	0.8586381
5	0.8725792	0.8042784
6	0.8781431	0.9352381
⋮	⋮	⋮
86398	0.814165	0.8291982
86399	0.8728324	0.811507

(a) Valores gerados pela LSTM.

t	H(dstIP)	H(dstPort)
0	0.9306859	0.9520608
1	0.6459135	0.9473867
2	0.8725544	0.9402426
3	0.8725544	0.9402426
4	0.8725544	0.9402426
5	0.858636	0.9174064
6	0.8421633	0.9458035
⋮	⋮	⋮
86398	0.8201721	0.9226764
86399	0.8544536	0.9146305

(b) Conjunto de testes.

Na tabela 14 é possível observar a validação do tráfego a partir do cálculo do NMSE e da correlação de Pearson.

Tabela 14 – Resultado das métricas de validação para as dimensões analisadas.

	H(dstIP)	H(dstPort)
NMSE	0.0099	0.0012
Pearson	0.1742	-0.0352

Desta forma, é possível concluir que, embora ambas as dimensões apresentam resultados satisfatórios, a previsão para o conjunto das entropias de portas de destino foi melhor do que a das entropias de IP de destino, devido ao valor do erro ser mais baixo. Além do mais, os valores da correlação de Pearson não são tão bons devido à grande quantidade de elementos analisados nas amostras.

7 RESULTADOS OBTIDOS - GRU *VS* LSTM

Com o intuito de realizar uma comparação dos resultados de mais de um método, foi realizada a implementação de um modelo de rede neural GRU, cujo treinamento e detecção de anomalias ocorre de maneira idêntica aos do LSTM, apresentados nas seções 5.1 e 5.1.1, respectivamente.

Os resultados da rede LSTM apresentados neste capítulo foram obtidos a partir do mesmo modelo de 6 neurônios e 30 épocas apresentado na tabela 9, entretanto, o modelo em questão é aplicado em um outro estudo de caso que será apresentado mais a frente, na seção 7.2. Os resultados da técnica GRU foram obtidos a partir da realização dos mesmos testes realizados pela LSTM no capítulo 5.

7.1 Treinamento e Validação

O treinamento da rede neural foi realizado para um conjunto de 1 à 10 neurônios, de forma que cada caso foi treinado com $5, 10, 15, 20, 25$ e 30 épocas; os resultados podem ser observados na tabela 15. Este método de treinamento foi escolhido para poder determinar qual é a combinação neurônio-época que gera a melhor acurácia ao terminar o treinamento da rede neural. Por fim, ao analisar a tabela das acurácias é possível concluir que o melhor resultado foi obtido para 5 neurônios e 25 épocas, produzindo uma acurácia de 98.14%, e o pior resultado foi obtido para 1 neurônio e 20 épocas, com 97.56% de acurácia.

Tabela 15 – Acurácia obtida em função de Neurônios × Épocas

	Épocas					
Neurônios	5	10	15	20	25	30
1	97.77%	97.74%	97.76%	**97.56%**	97.72%	97.75%
2	97.61%	97.74%	98.09%	97.94%	97.89%	98.07%
3	97.77%	97.90%	98.08%	97.91%	97.91%	98.05%
4	97.73%	97.75%	98.09%	97.94%	98.08%	98.04%
5	97.98%	98.04%	98.09%	98.12%	**98.14%**	98.05%
6	98.01%	98.10%	98.10%	98.11%	97.97%	97.97%
7	97.80%	98.01%	98.10%	98.09%	98.11%	98.03%
8	97.94%	98.08%	98.11%	98.03%	98.07%	98.10%
9	98.02%	98.07%	98.11%	98.05%	98.09%	98.02%
10	97.97%	98.07%	98.05%	98.11%	98.07%	98.06%

7.2 Detecção de anomalias

O processo de detecção de anomalias feito com os resultados produzidos pela rede GRU foi o mesmo realizado na seção 5.1.1. Sendo assim, foi analisada a divergência entre um tráfego sem anomalias, gerado pela GRU, com uma assinatura de tráfego contendo um ataque DDoS. A figura 48 mostra uma comparação entre três tipos de tráfego nas duas dimensões analisadas, representadas pelas colunas da figura, sendo que cada um dos gráficos simboliza 24 horas úteis de dados. Na primeira linha da figura consta um tráfego real, sem ataque, gerado pelo *Mininet*, utilizado como conjunto de testes da rede neural; na segunda linha, o tráfego gerado pelo GRU, sem ataque; e na terceira linha, um tráfego contendo uma anomalia de DDoS das 16:35 às 17:50 (intervalo destacado em vermelho).

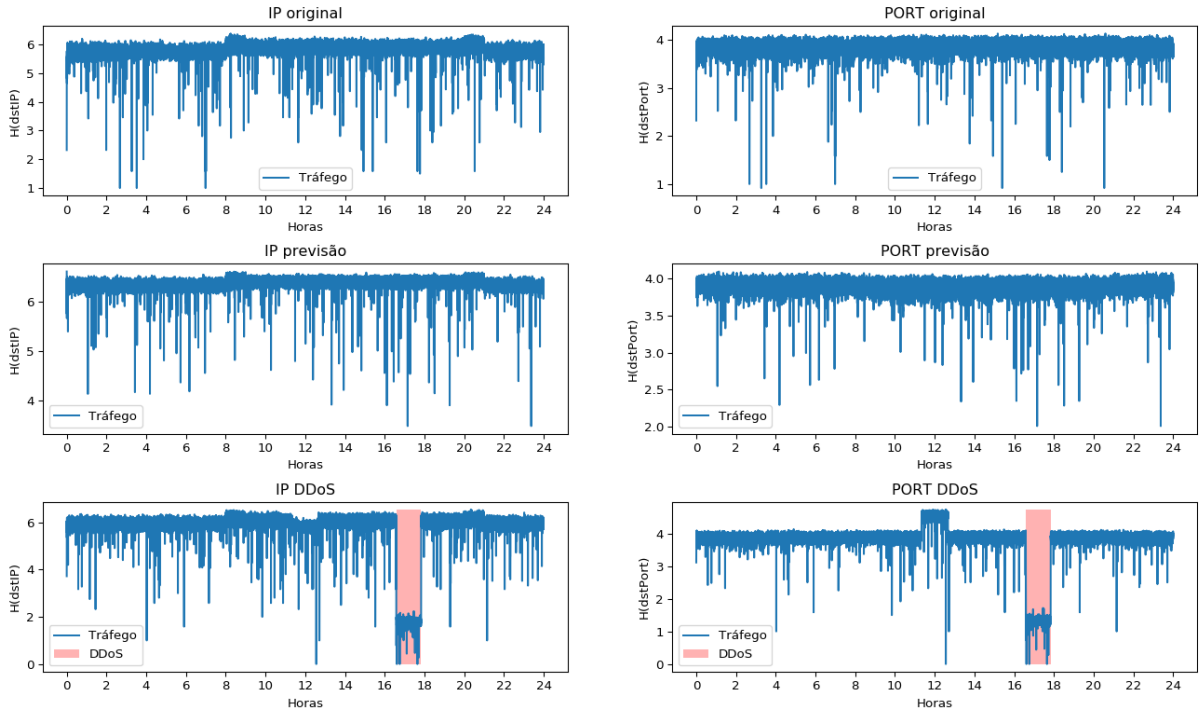


Figura 48 – Na primeira linha, os parâmetros originais, sem ataque; na segunda, os parâmetros gerados pela rede neural e na terceira linha, os parâmetros com DDoS. Fonte: próprio autor.

Para examinar a existência de uma anomalia de DDoS no tráfego analisado, foi aplicado um limiar superior e um limiar inferior. Em seguida, os valores foram aplicados no tráfego gerado pela rede neural, gerando intervalos de confiança que foram, então, comparados com um cenário anômalo.

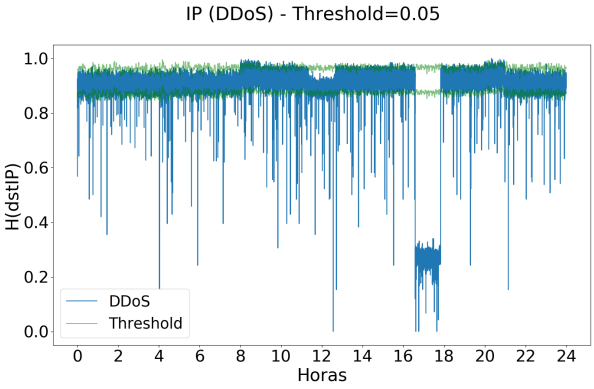
A fim de gerar os intervalos de confiança, o tráfego gerado pela rede GRU foi multiplicado por uma taxa de *threshold* em porcentagem, gerando, assim, dois conjuntos de dados. Depois, cada um deles foi divido em 60 partes iguais e foi calculada a média de cada uma das partes, gerando, assim, o intervalo de confiança final. Por fim, a matriz gerada pela rede neural foi concatenada lado a lado com a matriz das dimensões contendo

o DDoS, a matriz resultante foi normalizada entre 0 e 1 e depois separada novamente em matriz resultante e matriz de DDoS. Este processo tem como objetivo normalizar os intervalos de confiança em relação ao tráfego anômalo para o procedimento da detecção da anomalias.

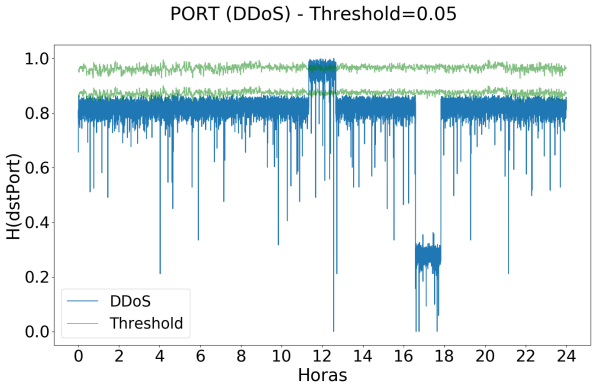
7.3 Resultados obtidos e comparação

Nesta seção, são apresentados os resultados obtidos pela rede GRU e serão comparados com os resultados obtidos pela rede LSTM; ambas em seus melhores casos de treinamento, aplicados no mesmo cenário anômalo.

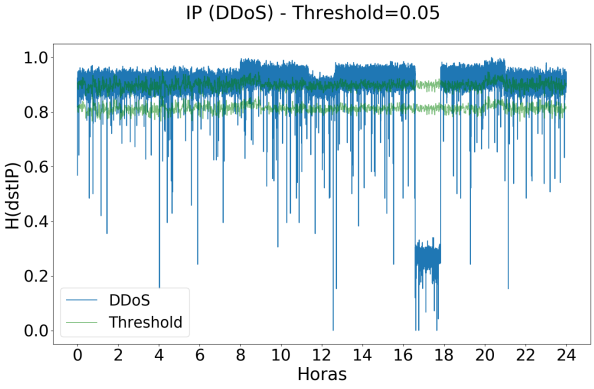
Para realizar a detecção da anomalia, pegou-se os dois conjuntos de limiares gerados pelo processo descrito na seção 7.2 e a comparação com o tráfego anômalo se deu de maneira que, todo o tráfego que estivesse contido dentro do intervalo de confiança era considerado normal e todo o tráfego que saía dele era considerado anômalo. Para encontrar os melhores limiares, foi feita a aplicação de taxas de *threshold* de 5% até 75%, incrementando valores de 5 em 5. Estes limiares e suas devidas comparações podem ser visualizados nas figuras 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44 e 45.



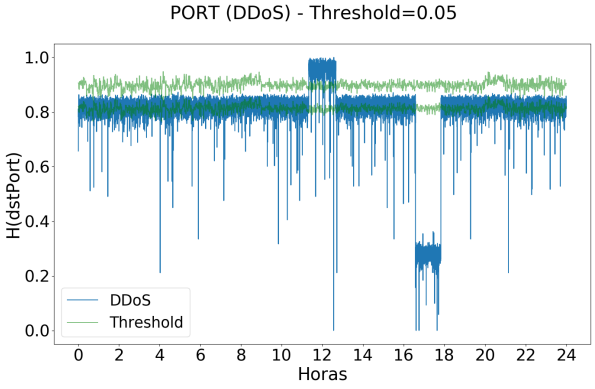
(a) Entropia de IP de destino - GRU.



(b) Entropia de portas destino - GRU.

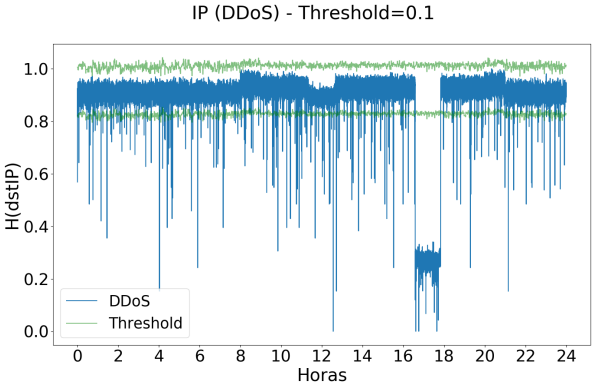


(c) Entropia de IP de destino - LSTM.

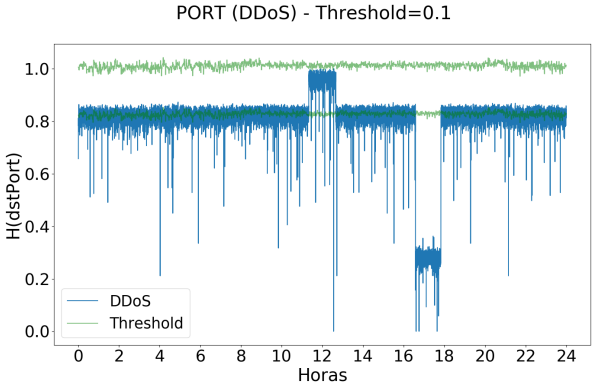


(d) Entropia de portas destino - LSTM.

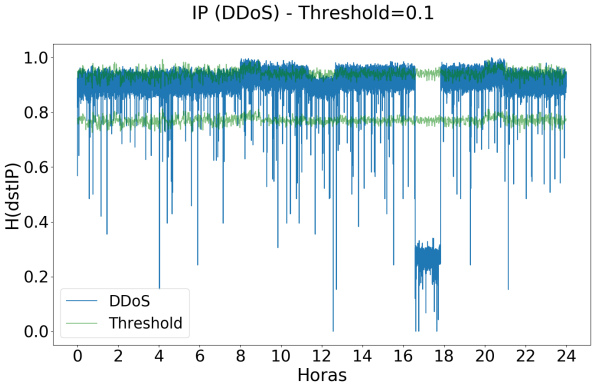
Figura 49 – Limiares de 5%, aplicados às dimensões analisadas. Fonte: próprio autor.



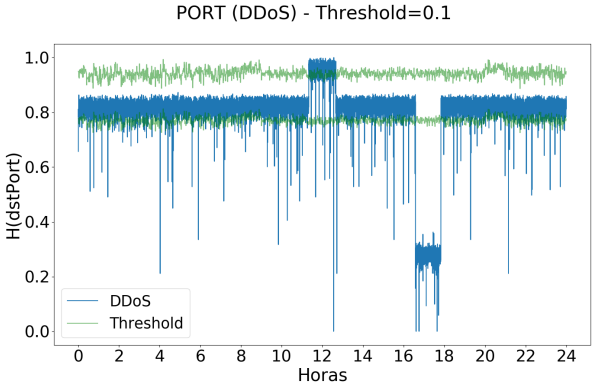
(a) Entropia de IP de destino - GRU.



(b) Entropia de portas destino - GRU.



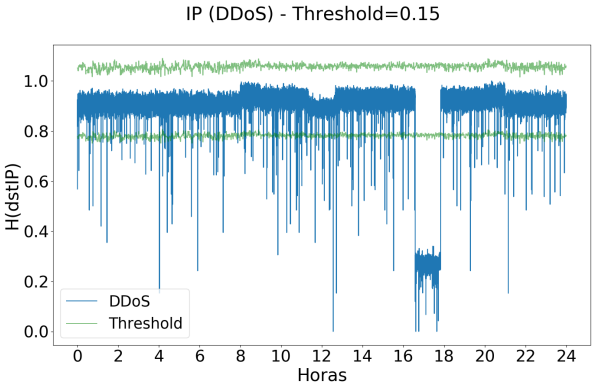
(c) Entropia de IP de destino - LSTM.



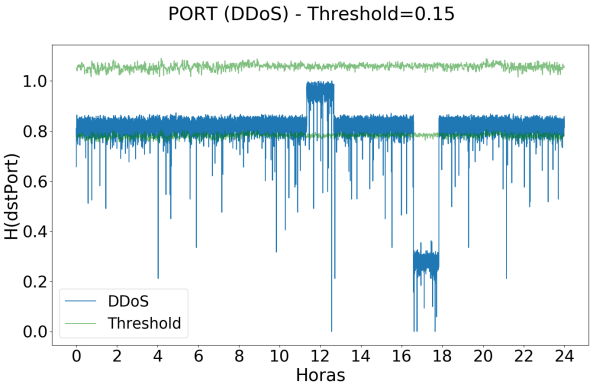
(d) Entropia de portas destino - LSTM.

Figura 50 – Limiares de 10%, aplicados às dimensões analisadas. Fonte: próprio autor.

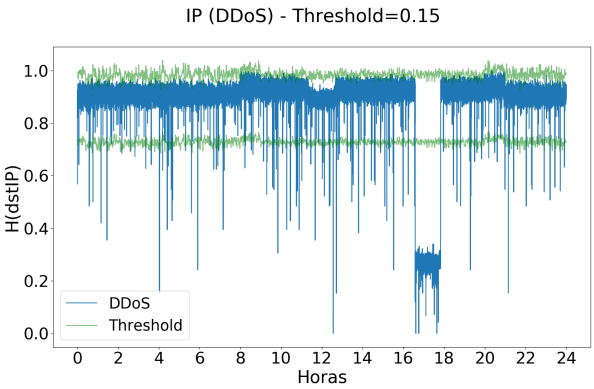
Os resultados das métricas foram calculados a partir dos valores de Verdadeiros Positivos (VP); Verdadeiros Negativos (VN); Falsos Positivos (FP); e Falsos Negativos (FN). Com base nestes resultados, foram calculadas as métricas de Precisão, Revocação, Acurácia e Taxa de Falsos Positivos, presentes nas tabelas 16 e 17.



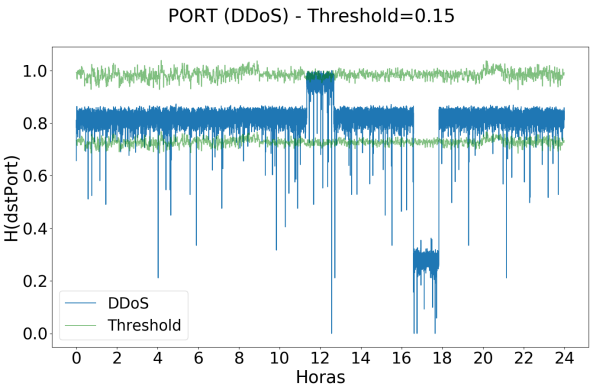
(a) Entropia de IP de destino - GRU.



(b) Entropia de portas destino - GRU.

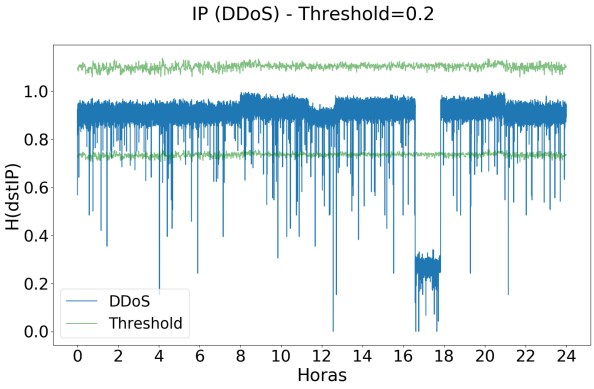


(c) Entropia de IP de destino - LSTM.

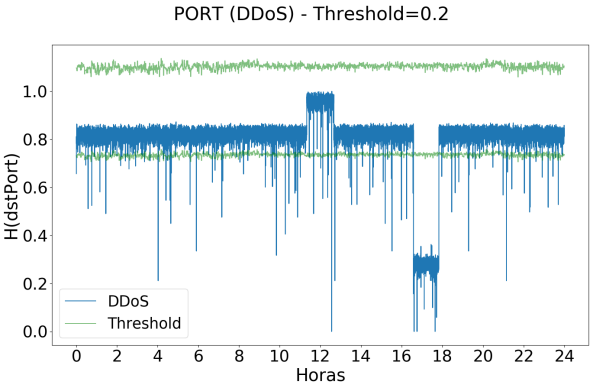


(d) Entropia de portas destino - LSTM.

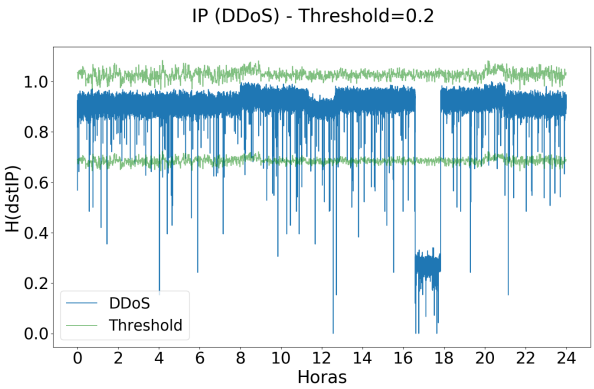
Figura 51 – Limiares de 15%, aplicados às dimensões analisadas. Fonte: próprio autor.



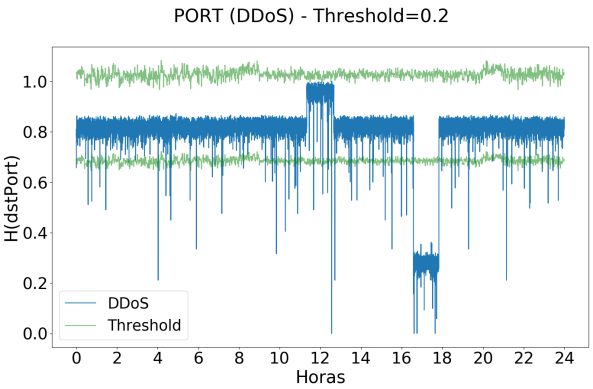
(a) Entropia de IP de destino - GRU.



(b) Entropia de portas destino - GRU.

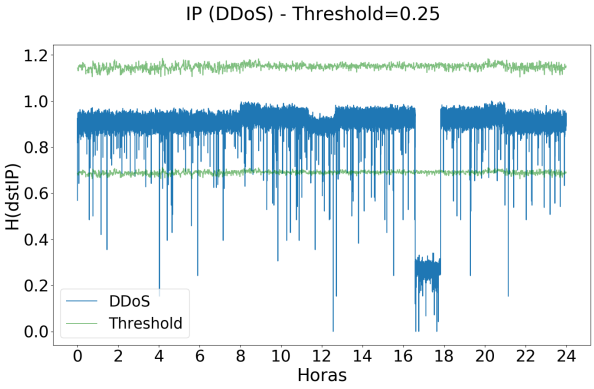


(c) Entropia de IP de destino - LSTM.

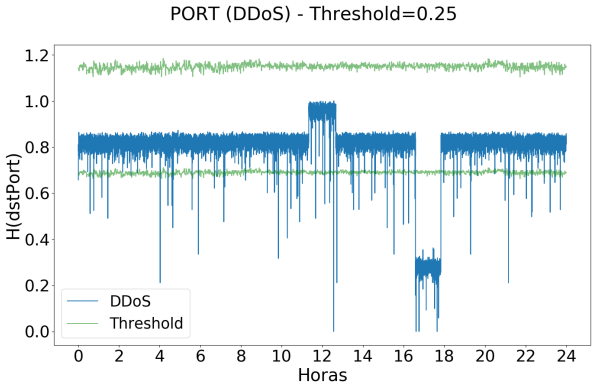


(d) Entropia de portas destino - LSTM.

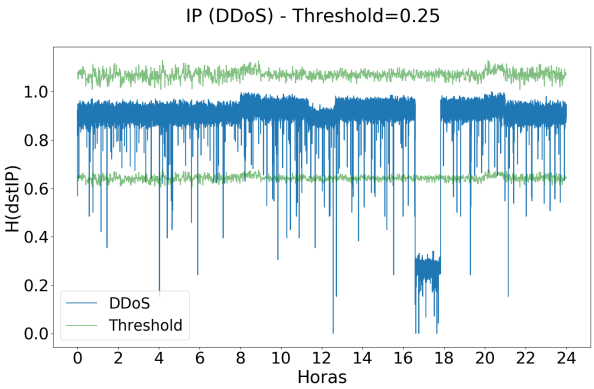
Figura 52 – Limiares de 20%, aplicados às dimensões analisadas. Fonte: próprio autor.



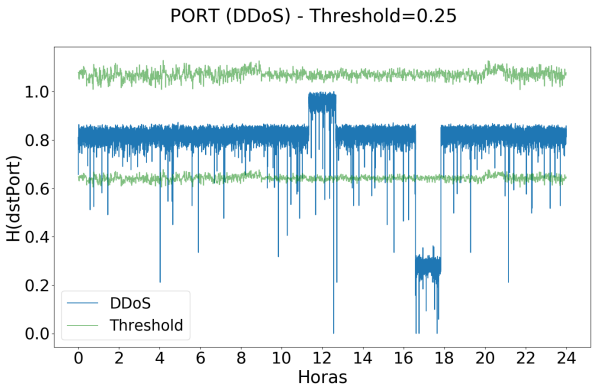
(a) Entropia de IP de destino - GRU.



(b) Entropia de portas destino - GRU.

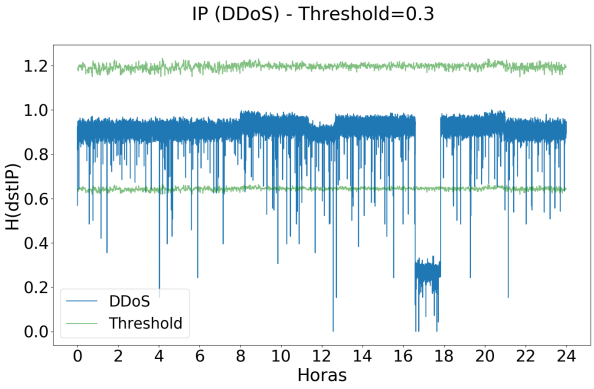


(c) Entropia de IP de destino - LSTM.

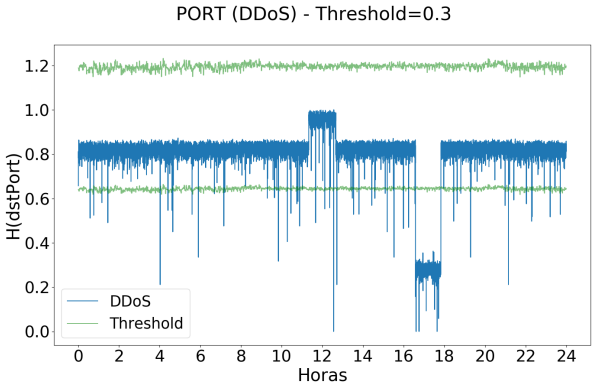


(d) Entropia de portas destino - LSTM.

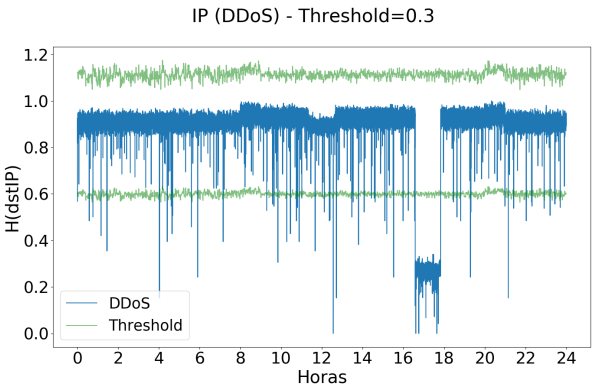
Figura 53 – Limiares de 25%, aplicados às dimensões analisadas. Fonte: próprio autor.



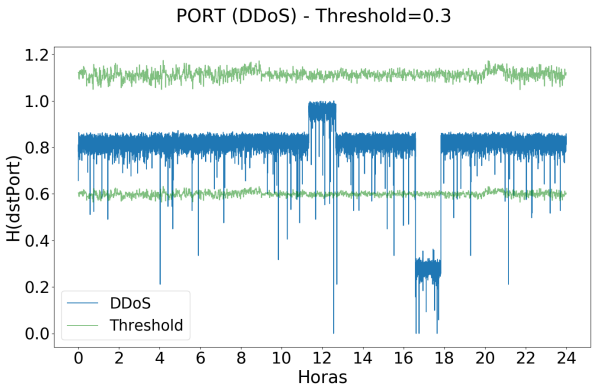
(a) Entropia de IP de destino - GRU.



(b) Entropia de portas destino - GRU.

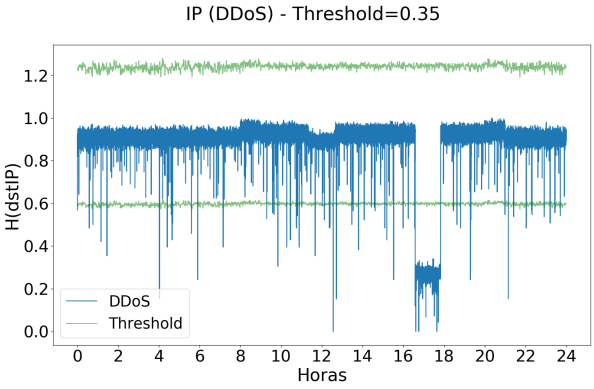


(c) Entropia de IP de destino - LSTM.

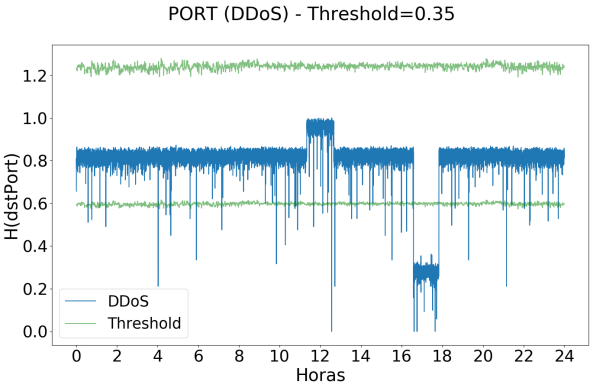


(d) Entropia de portas destino - LSTM.

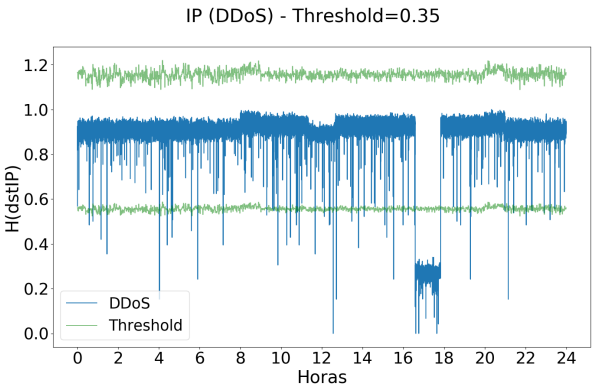
Figura 54 – Limiares de 30%, aplicados às dimensões analisadas. Fonte: próprio autor.



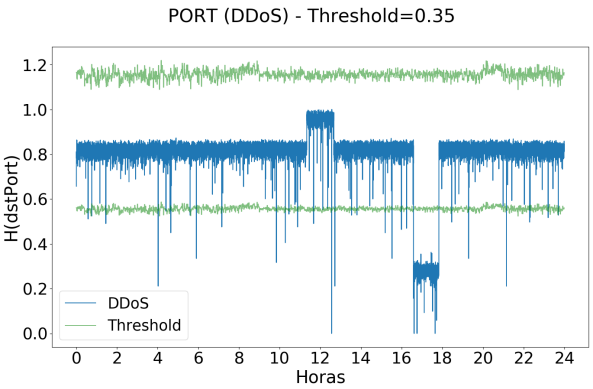
(a) Entropia de IP de destino - GRU.



(b) Entropia de portas destino - GRU.

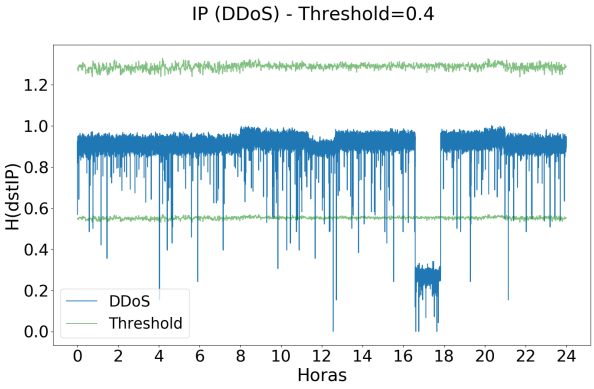


(c) Entropia de IP de destino - LSTM.

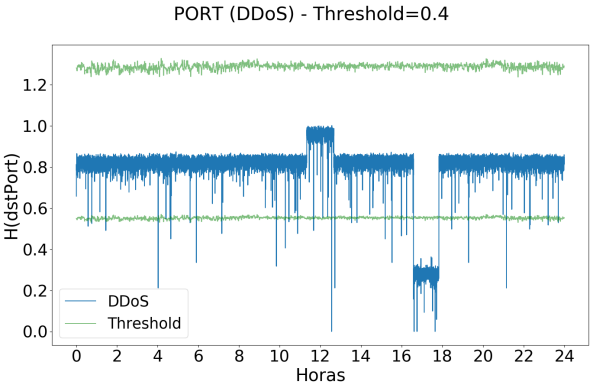


(d) Entropia de portas destino - LSTM.

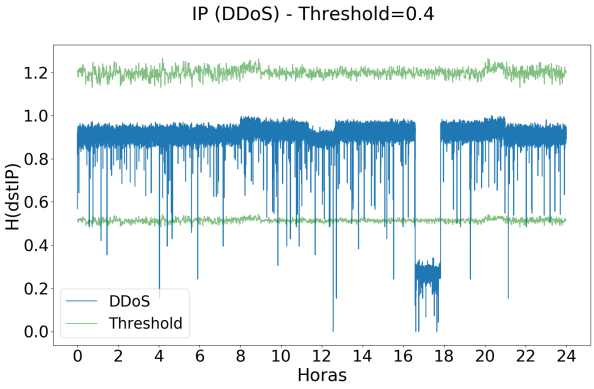
Figura 55 – Limiares de 35%, aplicados às dimensões analisadas. Fonte: próprio autor.



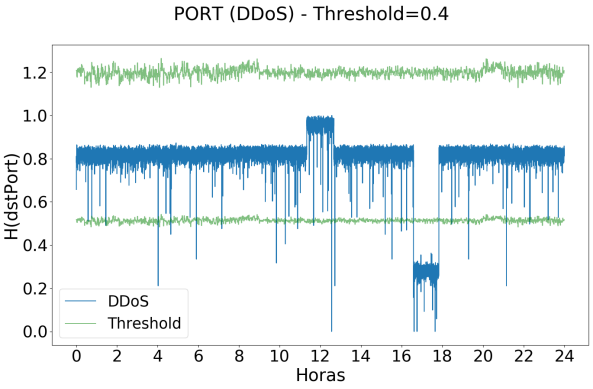
(a) Entropia de IP de destino - GRU.



(b) Entropia de portas destino - GRU.

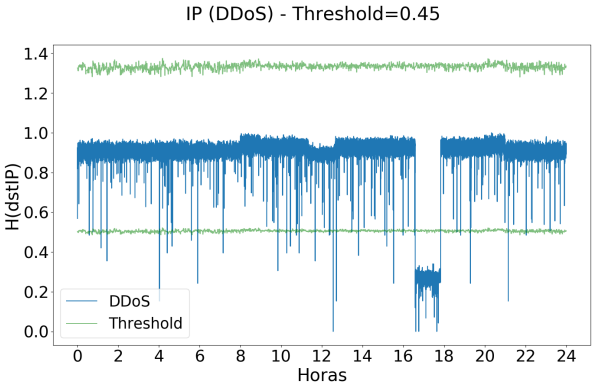


(c) Entropia de IP de destino - LSTM.

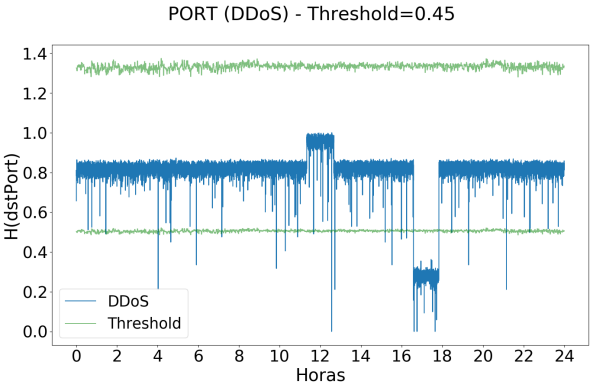


(d) Entropia de portas destino - LSTM.

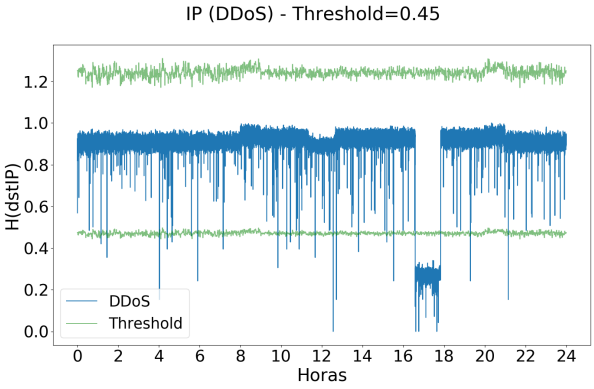
Figura 56 – Limiares de 40%, aplicados às dimensões analisadas. Fonte: próprio autor.



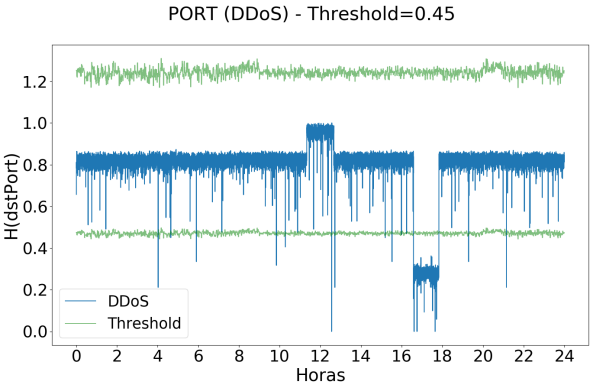
(a) Entropia de IP de destino - GRU.



(b) Entropia de portas destino - GRU.

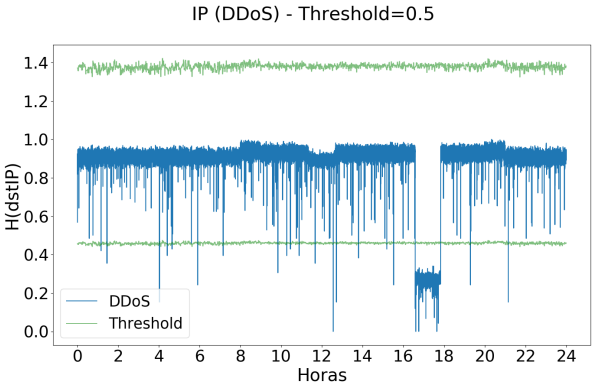


(c) Entropia de IP de destino - LSTM.

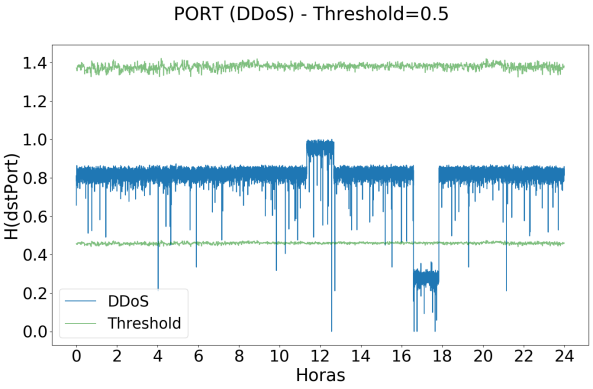


(d) Entropia de portas destino - LSTM.

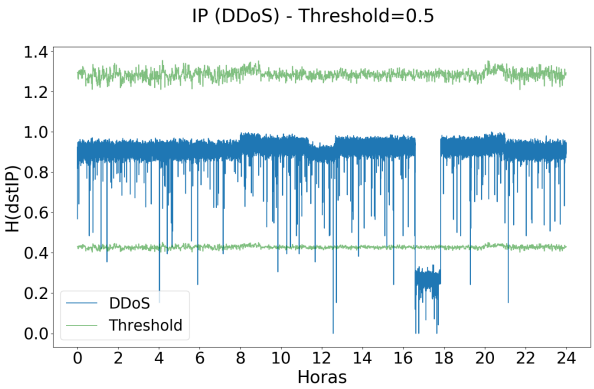
Figura 57 – Limiares de 45%, aplicados às dimensões analisadas. Fonte: próprio autor.



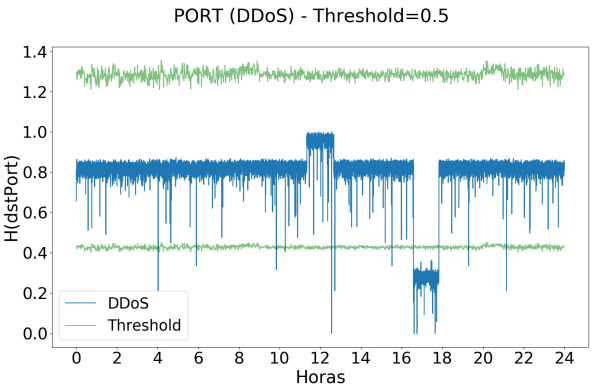
(a) Entropia de IP de destino - GRU.



(b) Entropia de portas destino - GRU.

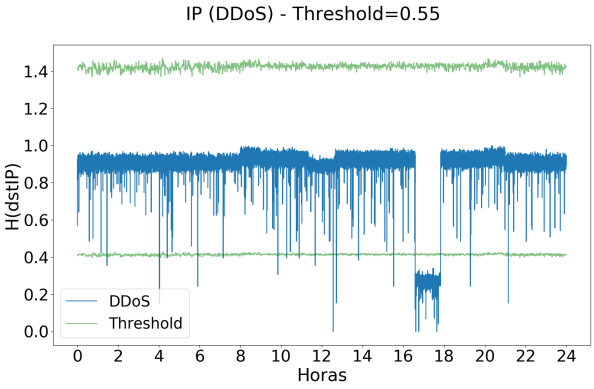


(c) Entropia de IP de destino - LSTM.

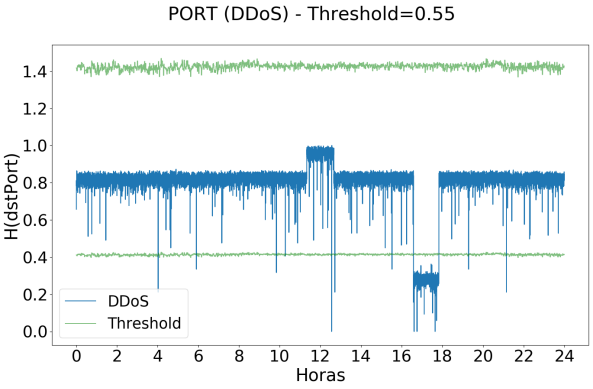


(d) Entropia de portas destino - LSTM.

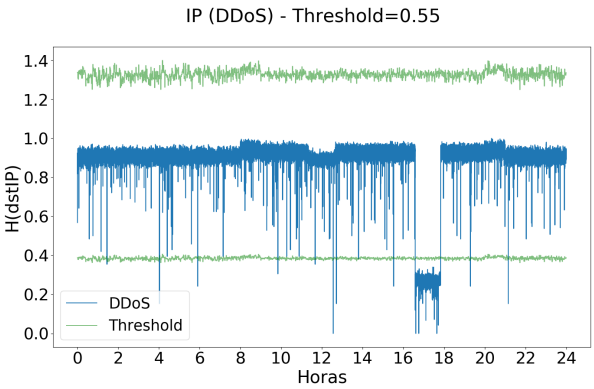
Figura 58 – Limiares de 50%, aplicados às dimensões analisadas. Fonte: próprio autor.



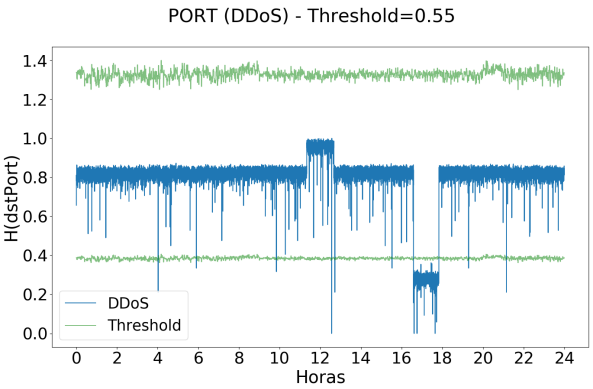
(a) Entropia de IP de destino - GRU.



(b) Entropia de portas destino - GRU.

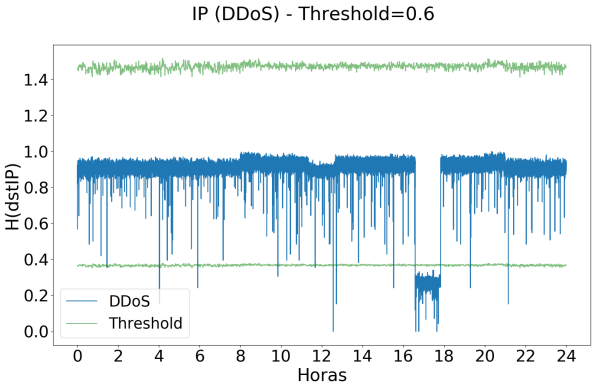


(c) Entropia de IP de destino - LSTM.

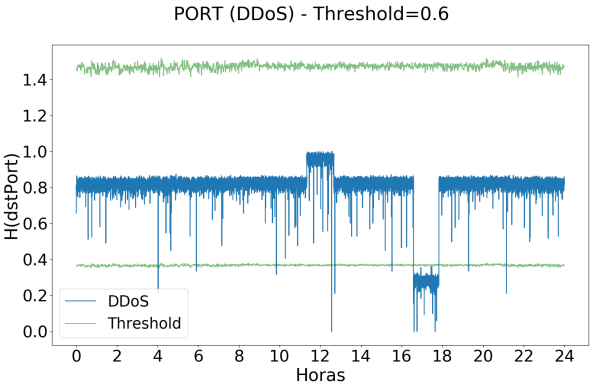


(d) Entropia de portas destino - LSTM.

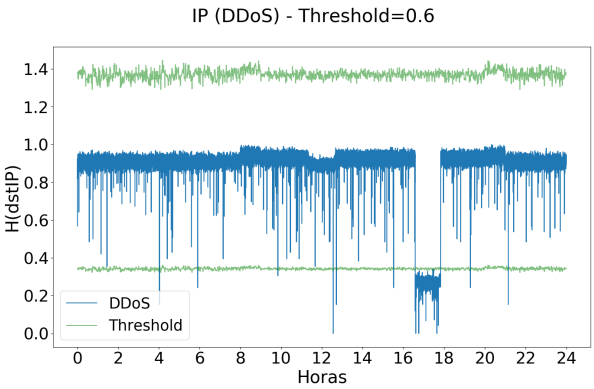
Figura 59 – Limiares de 55%, aplicados às dimensões analisadas. Fonte: próprio autor.



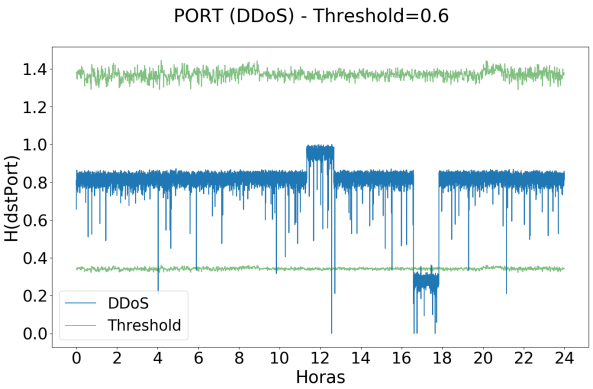
(a) Entropia de IP de destino - GRU.



(b) Entropia de portas destino - GRU.

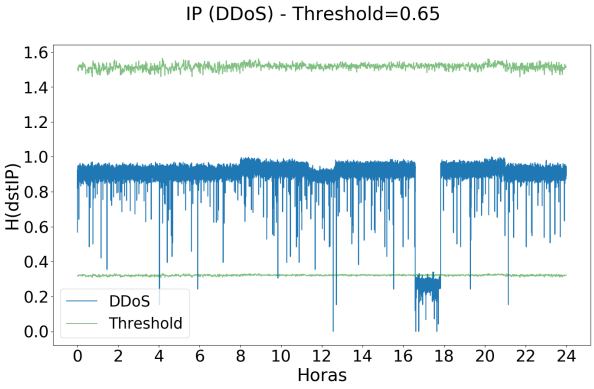


(c) Entropia de IP de destino - LSTM.

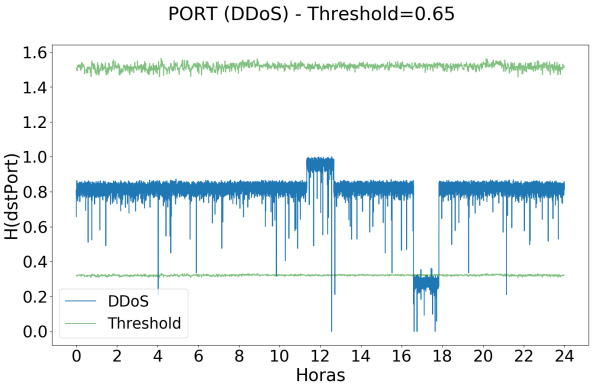


(d) Entropia de portas destino - LSTM.

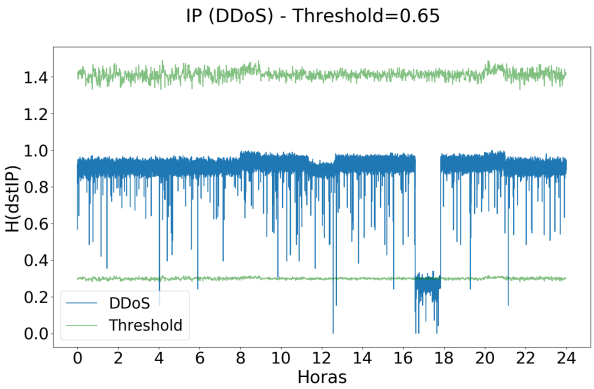
Figura 60 – Limiares de 60%, aplicados às dimensões analisadas. Fonte: próprio autor.



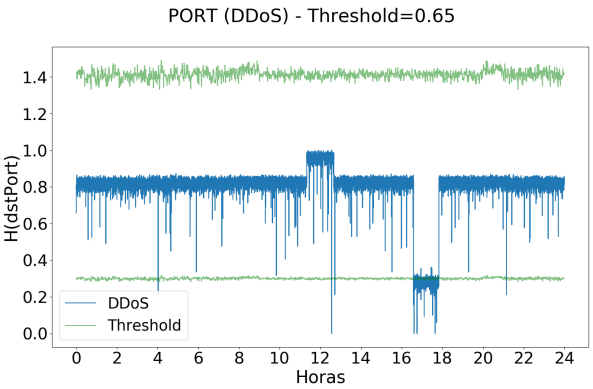
(a) Entropia de IP de destino - GRU.



(b) Entropia de portas destino - GRU.

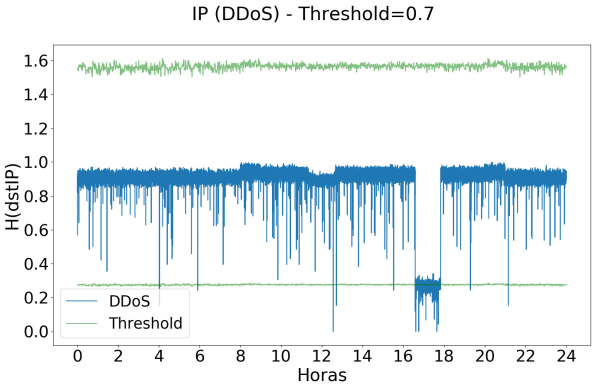


(c) Entropia de IP de destino - LSTM.

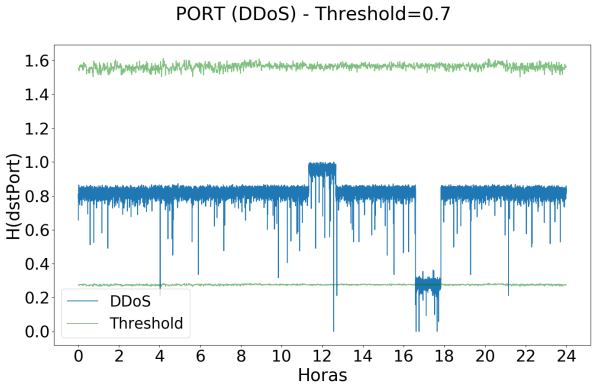


(d) Entropia de portas destino - LSTM.

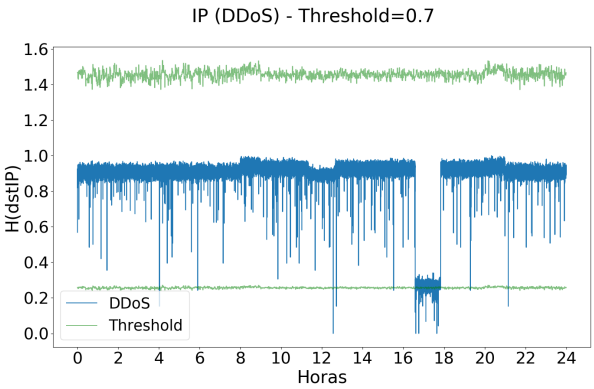
Figura 61 – Limiares de 65%, aplicados às dimensões analisadas. Fonte: próprio autor.



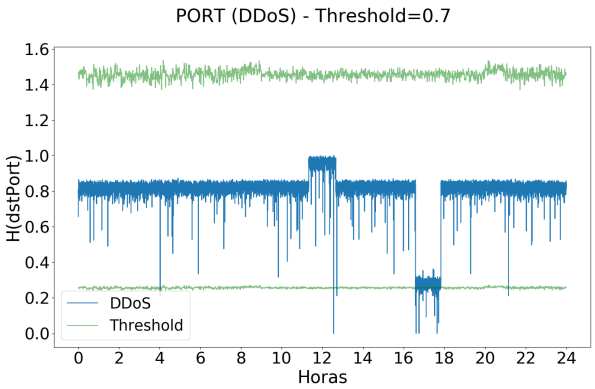
(a) Entropia de IP de destino - GRU.



(b) Entropia de portas destino - GRU.

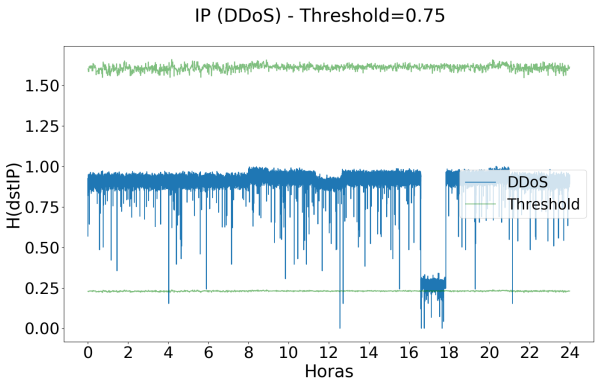


(c) Entropia de IP de destino - LSTM.

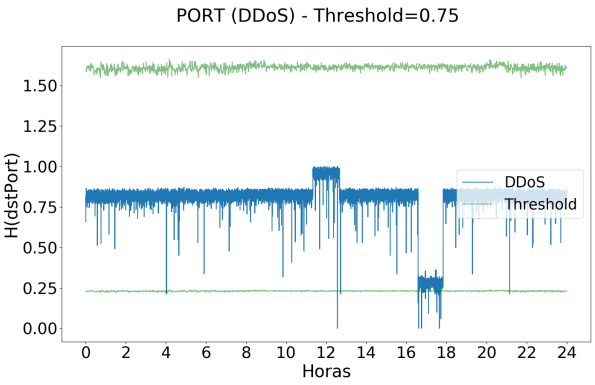


(d) Entropia de portas destino - LSTM.

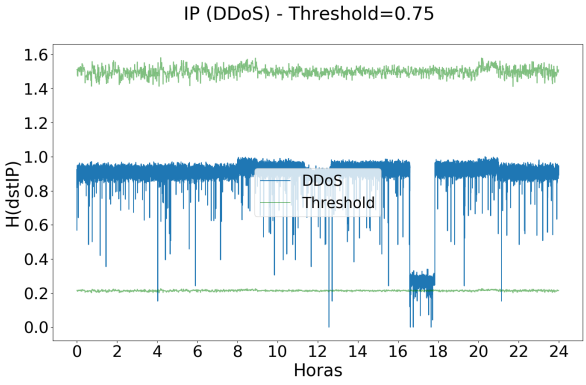
Figura 62 – Limiares de 70%, aplicados às dimensões analisadas. Fonte: próprio autor.



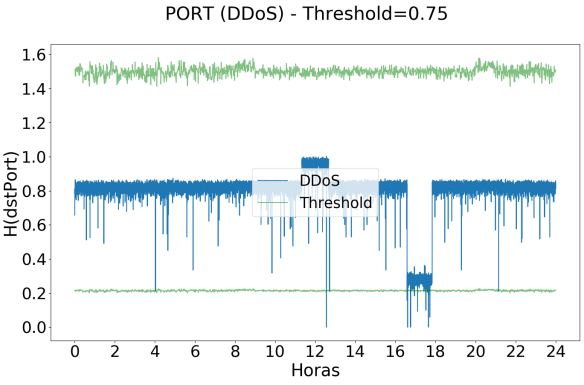
(a) Entropia de IP de destino - GRU.



(b) Entropia de portas destino - GRU.



(c) Entropia de IP de destino - LSTM.



(d) Entropia de portas destino - LSTM.

Figura 63 – Limiares de 75%, aplicados às dimensões analisadas. Fonte: próprio autor.

Tabela 16 – Comparação dos resultados das métricas de avaliação da detecção da anomalia de DDoS, para o conjunto de entropias de IP de destino.

	Métricas			
Limiares	Precisão	Revocação	Acurácia	Taxa de f.p.
5%	44.75%	100%	93.57%	6.78%
10%	94.24%	100%	99.68%	0.34%
15%	96.24%	100%	99.80%	0.21%
20%	97.19%	100%	99.85%	0.16%
25%	97.80%	100%	99.88%	0.12%
30%	98.23%	100%	99.91%	0.10%
35%	98.68%	100%	99.93%	0.07%
40%	98.92%	100%	99.94%	0.06%
45%	99.25%	100%	99.96%	0.04%
50%	99.56%	100%	99.98%	0.02%
55%	99.67%	100%	99.98%	0.02%
60%	99.78%	100%	99.99%	0.01%
65%	99.82%	99.93%	99.99%	0.01%
70%	99.82%	84.53%	99.19%	0.01%
75%	92.31%	1.07%	94.84%	0.00%

(a) Resultado das métricas - GRU.

	Métricas			
Limiares	Precisão	Revocação	Acurácia	Taxa de f.p.
5%	6.08%	100%	19.60%	84.82%
10%	17.27%	100%	75.05%	26.32%
15%	81.32%	100%	98.80%	1.26%
20%	97.80%	100%	99.88%	0.12%
25%	98.43%	100%	99.92%	0.09%
30%	98.71%	100%	99.93%	0.07%
35%	98.94%	100%	99.94%	0.06%
40%	99.25%	100%	99.96%	0.04%
45%	99.56%	100%	99.98%	0.02%
50%	99.62%	100%	99.98%	0.02%
55%	99.76%	100%	99.99%	0.01%
60%	99.82%	100%	99.99%	0.01%
65%	99.84%	98.42%	99.91%	0.01%
70%	99.44%	27.47%	96.21%	0.01%
75%	88.24%	0.67%	94.82%	0.00%

(b) Resultado das métricas - LSTM.

Tabela 17 – Comparação dos resultados das métricas de avaliação da detecção da anomalia de DDoS, para o conjunto de entropias de portas de destino.

	Métricas			
Limiares	Precisão	Revocação	Acurácia	Taxa de f.p.
5%	5.51%	100%	10.75%	94.15%
10%	12.72%	100%	64.27%	37.69%
15%	68.51%	100%	97.61%	2.53%
20%	97.80%	100%	99.88%	0.12%
25%	98.55%	100%	99.92%	0.08%
30%	98.86%	100%	99.94%	0.06%
35%	99.14%	100%	99.95%	0.05%
40%	99.38%	100%	99.97%	0.03%
45%	99.62%	100%	99.98%	0.02%
50%	99.78%	100%	99.99%	0.01%
55%	99.82%	100%	99.99%	0.01%
60%	99.82%	100%	99.99%	0.01%
65%	99.91%	99.44%	99.97%	0.00%
70%	99.77%	37.76%	96.75%	0.00%
75%	84.62%	0.49%	94.81%	0.00%

(a) Resultado das métricas - GRU.

	Métricas			
Limiares	Precisão	Revocação	Acurácia	Taxa de f.p.
5%	11.03%	100%	57.98%	44.33%
10%	32.65%	100%	89.26%	11.33%
15%	68.83%	100%	97.64%	2.49%
20%	95.18%	100%	99.74%	0.28%
25%	98.81%	100%	99.94%	0.07%
30%	99.12%	100%	99.95%	0.05%
35%	99.34%	100%	99.97%	0.04%
40%	99.49%	100%	99.97%	0.03%
45%	99.71%	100%	99.98%	0.02%
50%	99.78%	100%	99.99%	0.01%
55%	99.82%	100%	99.99%	0.01%
60%	99.84%	99.93%	99.99%	0.01%
65%	99.90%	91.49%	99.55%	0.00%
70%	98.77%	7.13%	95.16%	0.00%
75%	86.67%	0.29%	94.80%	0.00%

(b) Resultado das métricas - LSTM.

8 ANÁLISE DOS RESULTADOS - GRU *VS* LSTM

Este capítulo, de forma semelhante ao capítulo 6, tem como objetivo analisar os resultados apresentados no capítulo 7, com o intuito de apresentar qual foi o melhor e o pior cenário de treinamento da rede neural, e quais foram os melhores limiares utilizados na detecção de anomalias e compara lado a lado os resultados da rede neural GRU com a rede neural LSTM. Além disso, serão apresentadas tabelas com a validação da predição das dimensões de entropia de IP de destino e entropia de portas de destino para ambos os modelos de rede neural implementados, utilizando as métricas de NMSE e a Correlação de Pearson.

8.1 Cenários de treinamento

Os cenários de treinamento da rede neural GRU foram os mesmos da rede neural LSTM, descritos na seção 6.1. Sendo assim, foi feita a análise da tabela 15, descrita na seção 7.1, que compara os resultados dos treinamentos a partir do uso da métrica de acurácia aplicada em cada uma das combinações de neurônios e épocas. A combinação de treinamento (Neurônios × Épocas) que obteve a melhor acurácia em relação a todos os outros casos, foi a configuração na qual são caracterizados 5 neurônios e 25 épocas. Na figura 64 é possível observar o valor da acurácia em função durante o treinamento deste caso.

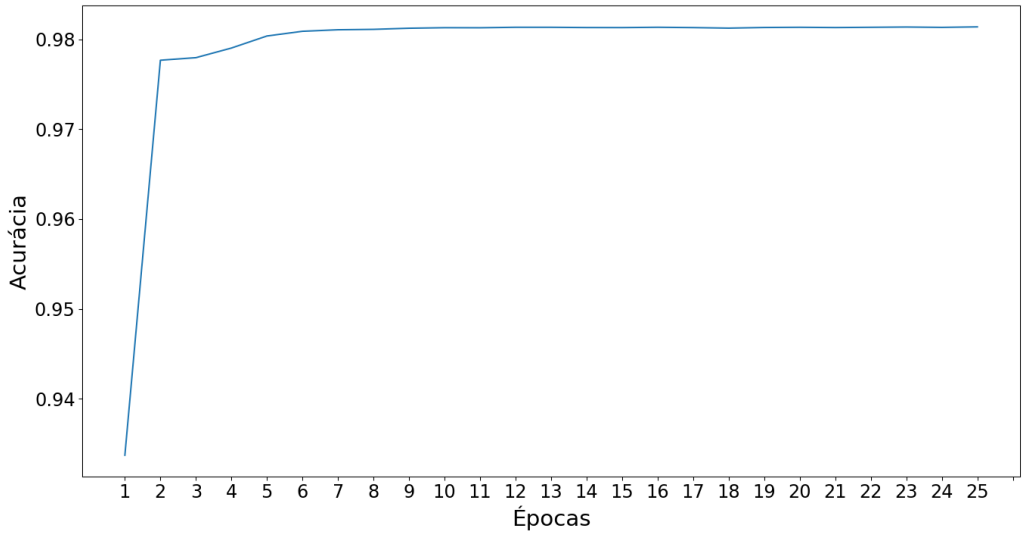


Figura 64 – Evolução da acurácia para 5 neurônios, em função do número de épocas. Fonte: próprio autor.

As figuras 27 e 48 foram geradas a partir do resultado da melhor combinação,

em que as figuras ilustram, respectivamente, a rede neural implementada e a comparação entre o tráfego gerado com um tráfego real não anômalo e um tráfego real contendo uma anomalia de DDoS. Da mesma forma, foram geradas as tabelas 16a e 17a, que ilustram, respectivamente, os resultados das métricas calculadas para as dimensões de entropia de IP de destino e entropia de portas de destino para a rede GRU.

8.2 Melhores limiares na detecção de anomalias

Considerando as tabelas 16 e 17, pode-se inferir que em ambos os modelos de RNN, os resultados de detecção de anomalias são melhores na dimensão de entropias de IP de destino, pois apresentam maior precisão para menores intervalos de confiança. Além disso é possível concluir que para a rede neural GRU foi possível gerar um intervalo de confiança melhor do que o gerado a partir da LSTM nesta dimensão, já que para um *threshold* de 10% a precisão da GRU é igual a 94.24%, enquanto que, para a LSTM atingir uma precisão maior do que 90% foi necessário um intervalo de confiança de ao menos 20%. Ao analisar as figuras 49a e 50a, que representam os limiares de 5% e 10% na dimensão das entropias de ip de destino, é possível notar que o limiar se encaixa muito bem no tráfego anômalo. Entretanto, na dimensão das entropias de portas de destino, ao observar as figuras 49b, 50b, 51b, 52b, 53b, 54b, 55b, 56b, 57b, 58b, 59b, 60b, 61b, 62b e 63b, além da tabela 17, pode-se notar que o GRU e o LSTM tiveram uma convergência similar, sendo que ambos obtiveram precisão maior que 90% apenas com *threshold* maior do que 20%. Todavia, para esta dimensão, a rede LSTM obteve resultados um pouco melhores para a maioria dos intervalos de precisão. Ademais, ambos os modelos conseguem detectar todo o intervalo anômalo para os limiares de 5% até 60% em ambas as dimensões, a partir disso o tráfego anômalo é englobado pelo limiar e passa a ser considerado tráfego normal.

Ainda analisando as tabelas 16 e 17, embora a rede GRU tenha apresentado resultados melhores do que a rede LSTM para limiares menores, a maioria dos limiares acima de 20% e 25% obtiveram uma precisão um pouco melhor nas entropias de IP de destino no LSTM. E para a dimensão de entropia de portas de destino, a rede LSTM obteve uma performance geral melhor do que a GRU, exceto para o limiar de 20% e desconsiderando os limiares estritamente maiores do que 60%.

8.3 Validação da previsão

Esta seção tem como objetivo validar as previsões realizadas pela rede neural GRU em detrimento a rede neural LSTM. A seção está estruturada da mesma maneira que a seção 6.3, bem como as métricas propostas. Além disso, os dados das dimensões estudadas e do conjunto de testes estão normalizados num intervalo entre 0 e 1, como mostra a tabela 18. O cálculo foi realizado para cada uma das dimensões.

Tabela 18 – Matrizes dos conjuntos de predição e conjunto de testes.

t	H(dstIP)	H(dstPort)
0	1.0	0.9165649
1	0.7588066	0.952459
2	0.9509995	0.9138642
3	0.9517527	0.9108253
4	0.9524331	0.9086619
5	0.9422802	0.8750433
6	0.9247276	0.928038
⋮	⋮	⋮
86398	0.9032907	0.8897109
86399	0.9334274	0.8754405

(a) Valores gerados pela GRU.

t	H(dstIP)	H(dstPort)
0	0.9306859	0.9520608
1	0.6459135	0.9473867
2	0.8725544	0.9402426
3	0.8725544	0.9402426
4	0.8725544	0.9402426
5	0.858636	0.9174064
6	0.8421633	0.9458035
⋮	⋮	⋮
86398	0.8201721	0.9226764
86399	0.8544536	0.9146305

(b) Conjunto de testes.

Na tabela 19 é possível observar a validação do tráfego a partir do cálculo do NMSE e da correlação de Pearson com relação a ambas as redes neurais.

Tabela 19 – Comparação dos resultados das métricas de validação do tráfego gerado pela rede GRU e LSTM.

	H(dstIP)	H(dstPort)
NMSE	0.0107	0.0012
Pearson	0.2238	-0.0315

(a) Resultados - GRU.

	H(dstIP)	H(dstPort)
NMSE	0.0099	0.0012
Pearson	0.1742	-0.0352

(b) Resultados - LSTM.

Ao analisar a comparação das métricas de validação apresentada pela tabela 19 é possível concluir que existe uma variação muito pequena nas métricas de NMSE e Pearson para ambos os modelos de RNN implementados. Assim, pode-se dizer que, para as duas dimensões analisadas, ambas as redes neurais obtiveram resultados satisfatórios de predição, devido, principalmente, ao baixo erro. Os valores da correlação de Pearson não são tão bons devido a grande quantidade de elementos contidos nas amostras, entretanto, os resultados desta métrica também ficam bastante próximos para o tráfego gerado pela LSTM e pela GRU.

Finalmente, é possível concluir que ambos os métodos obtiveram resultados satisfatórios no quesito predição de tráfego e detecção de anomalias.

9 CONTRIBUIÇÕES E TRABALHOS FUTUROS

Este trabalho teve como premissa estudar métodos de *Deep Learning* e como são aplicados em detecção de anomalias em redes; assunto no qual ainda não havia sido abordado pelo grupo de pesquisa de redes de computadores do Departamento de Computação da Universidade Estadual de Londrina, que realiza pesquisas em gerência, segurança e detecção de intrusões e anomalias em redes de computadores. Conjuntamente, foi realizado um estudo de caso com o intuito de realizar a implementação de um método de aprendizado profundo que fosse capaz de caracterizar um tráfego de rede e, a partir disto, realizar a detecção de uma anomalia de DDoS.

A partir dos estudos realizados, é possível afirmar que os métodos de aprendizado de máquina, especialmente aprendizado profundo, tem uma vantagem significativa para a resolução dos mais diversos problemas computacionais nos dias de hoje, devido principalmente ao fato de que operam bem com grandes quantidades de informações.

No quesito detecção de anomalias em redes, os modelos de redes neurais profundas têm se mostrado apropriados, devido à capacidade dos métodos de identificar padrões, que é um ponto importante a se considerar no âmbito de segurança *online*. Isto graças à complexidade peculiar aos mais diversos tipos de anomalias existentes, de forma que algumas delas ainda apresentam desafios significativos aos pesquisadores e aos gerentes de rede.

Tendo em consideração os resultados obtidos neste trabalho, foi possível concluir que as redes neurais profundas implementadas, o LSTM e o GRU, obtiveram resultados adequados ao caracterizar o tráfego e durante o processo de detecção de anomalias, embora ainda seja necessária a realização de ajustes no modelo. Ademais, espera-se que a partir do estudo de caso realizado seja possível ter uma melhor compreensão do uso de redes neurais profundas aplicadas na detecção de anomalias em redes, além de um maior entendimento das dimensões de entropia, no que diz respeito à análise de tráfego de rede.

Como trabalho futuro, espera-se a realização de ajustes finos na rede neural, de forma que seja possível generalizar ainda mais a caracterização do tráfego, a fim de diminuir o tamanho do intervalo de confiança e detectar mais classes de anomalias. Além disso, uma possível mudança na implementação do modelo seria utilizar os dias de tráfego normal da base de dados de forma a partir do tempo zero da dimensão i_c de cada um dos conjuntos c de dimensões do tráfego, a fim de prever o tempo zero da dimensão i_p para o conjunto p, que indica o conjunto das dimensões de tráfego previsto pela rede neural. O modelo proposto se trata, portanto, de uma rede neural recorrente do tipo LSTM que utiliza o paradigma *muitos-para-um*, discutido na sessão 2.9.2, para que seja feita uma

comparação com os resultados obtidos neste trabalho.

REFERÊNCIAS

[1] WANG, H.; GU, J.; WANG, S. An effective intrusion detection framework based on svm with feature augmentation. *Knowledge-Based Systems*, v. 136, p. 130 – 139, 2017. ISSN 0950-7051. doi: 10.1016/j.knosys.2017.09.014.

[2] Shone, N. et al. A deep learning approach to network intrusion detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, v. 2, n. 1, p. 41–50, Feb 2018. ISSN 2471-285X. doi: 10.1109/TETCI.2017.2772792.

[3] Dong, B.; Wang, X. Comparison deep learning method to traditional methods using for network intrusion detection. In: *2016 8th IEEE International Conference on Communication Software and Networks (ICCSN)*. [S.l.: s.n.], 2016. p. 581–585. doi: 10.1109/ICCSN.2016.7586590.

[4] Kim, D. E.; Gofman, M. Comparison of shallow and deep neural networks for network intrusion detection. In: *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*. [S.l.: s.n.], 2018. p. 204–208. doi: 10.1109/CCWC.2018.8301755.

[5] FERNANDES, G. et al. A comprehensive survey on network anomaly detection. *Telecommunication Systems*, v. 70, n. 3, p. 447–489, Mar 2019. ISSN 1572-9451. doi: 10.1007/s11235-018-0475-8.

[6] Nagpal, B. et al. Ddos tools: Classification, analysis and comparison. In: *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*. [s.n.], 2015. p. 342–346. Disponível em: <https://ieeexplore.ieee.org/document/7100270>.

[7] Karatas, G.; Demir, O.; Koray Sahingoz, O. Deep learning in intrusion detection systems. In: *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*. [S.l.: s.n.], 2018. p. 113–116. doi: 10.1109/IBIGDELFT.2018.8625278.

[8] Borkar, A.; Donode, A.; Kumari, A. A survey on intrusion detection system (ids) and internal intrusion detection and protection system (iidps). In: *2017 International Conference on Inventive Computing and Informatics (ICICI)*. [S.l.: s.n.], 2017. p. 949–953. doi: 10.1109/ICICI.2017.8365277.

[9] Zhang, W.; Yang, Q.; Geng, Y. A survey of anomaly detection methods in networks. In: *2009 International Symposium on Computer Network and Multimedia Technology*. [S.l.: s.n.], 2009. p. 1–3. doi: 10.1109/CNMT.2009.5374676.

[10] AMARAL, A. A. et al. Deep ip flow inspection to detect beyond network anomalies. *Computer Communications*, v. 98, p. 80 – 96, 2017. ISSN 0140-3664. doi: 10.1016/j.comcom.2016.12.007.

[11] Yi Yuan; Hoong Kee Ng. Datcons: Protecting web-based qos from ddos attacks. In: *2006 IEEE/IFIP Network Operations and Management Symposium NOMS 2006*. [S.l.: s.n.], 2006. p. 1–4. doi: 10.1109/NOMS.2006.1687636.

[12] Suriadi, S.; Clark, A.; Schmidt, D. Validating denial of service vulnerabilities in web services. In: *2010 Fourth International Conference on Network and System Security*. [S.l.: s.n.], 2010. p. 175–182. doi: 10.1109/NSS.2010.41.

[13] Piskozub, A. Denial of service and distributed denial of service attacks. In: *Modern Problems of Radio Engineering, Telecommunications and Computer Science (IEEE Cat. No.02EX542)*. [S.l.: s.n.], 2002. p. 303–304. doi: 10.1109/TCSET.2002.1015977.

[14] Thomas, R. M.; James, D. Ddos detection and denial using third party application in sdn. In: *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*. [S.l.: s.n.], 2017. p. 3892–3897. doi: 10.1109/ICECDS.2017.8390193.

[15] Vormayr, G.; Zseby, T.; Fabini, J. Botnet communication patterns. *IEEE Communications Surveys Tutorials*, v. 19, n. 4, p. 2768–2796, Fourthquarter 2017. doi: 10.1109/COMST.2017.2749442.

[16] ZARPELãO, B. B. et al. A survey of intrusion detection in internet of things. *Journal of Network and Computer Applications*, v. 84, p. 25 – 37, 2017. ISSN 1084-8045. doi: 10.1016/j.jnca.2017.02.009.

[17] Daud, M. et al. Denial of service: (dos) impact on sensors. In: *2018 4th International Conference on Information Management (ICIM)*. [S.l.: s.n.], 2018. p. 270–274. doi: 10.1109/INFOMAN.2018.8392848.

[18] Gopal, T. S. et al. Mitigating mirai malware spreading in iot environment. In: *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. [S.l.: s.n.], 2018. p. 2226–2230. doi: 10.1109/ICACCI.2018.8554643.

[19] Sinanović, H.; Mrdovic, S. Analysis of mirai malicious software. In: *2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. [S.l.: s.n.], 2017. p. 1–5. doi: 10.23919/SOFTCOM.2017.8115504.

[20] WHAT is a DDoS Attack? <https://www.f5.com/labs/articles/education/what-is-a-distributed-denial-of-service-attack->. Acessado: 2019-01-12.

[21] Kreutz, D. et al. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, v. 103, n. 1, p. 14–76, Jan 2015. ISSN 0018-9219. doi: 10.1109/JPROC.2014.2371999.

[22] Kalkan, K.; Gur, G.; Alagoz, F. Defense mechanisms against ddos attacks in sdn environment. *IEEE Communications Magazine*, v. 55, n. 9, p. 175–179, Sep. 2017. ISSN 0163-6804. doi: 10.1109/MCOM.2017.1600970.

[23] Prajapati, A.; Sakadasariya, A.; Patel, J. Software defined network: Future of networking. In: *2018 2nd International Conference on Inventive Systems and Control (ICISC)*. [S.l.: s.n.], 2018. p. 1351–1354. doi: 10.1109/ICISC.2018.8399028.

[24] Olimjonovich, M. S. Software defined networking: Management of network resources and data flow. In: *2016 International Conference on Information Science and Communications Technologies (ICISCT)*. [S.l.: s.n.], 2016. p. 1–3. ISSN null. doi: 10.1109/ICISCT.2016.7777384.

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

[26] PAKZAD, F. et al. Efficient topology discovery in openflow-based software defined networks. *Computer Communications*, v. 77, p. 52 – 61, 2016. ISSN 0140-3664. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0140366415003527>.

[27] AHMED, M.; MAHMOOD, A. N.; HU, J. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, v. 60, p. 19 – 31, 2016. ISSN 1084-8045. doi: 10.1016/j.jnca.2015.11.016.

[28] LYRA, M. R. *Governança da Segurança da Informação*. 1. ed. Brasília: Mauricio Rocha Lyra, 2015. v. 1. ISBN 978-85-920264-1-7.

[29] FERNANDES, G. et al. Statistical, forecasting and metaheuristic techniques for network anomaly detection. In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. New York, NY, USA: ACM, 2015. (SAC '15), p. 701–707. ISBN 978-1-4503-3196-8. doi: 10.1145/2695664.2695852.

[30] Carvalho, L. F. et al. A novel anomaly detection system to assist network management in sdn environment. In: *2017 IEEE International Conference on Communications (ICC)*. [S.l.: s.n.], 2017. p. 1–6. doi: 10.1109/ICC.2017.7997214.

[31] 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.

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

[33] SAMRIN, R.; VASUMATHI, D. Review on anomaly based network intrusion detection system. *2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)*, p. 141–147, 2017.

[34] JR., M. L. P. et al. Digital signature to help network management using flow analysis. *Int. Journal of Network Management*, v. 26, n. 2, p. 76–94, 2016. doi: 10.1002/nem.1892.

[35] De Assis, M. V. O. et al. A game theoretical based system using holt-winters and genetic algorithm with fuzzy logic for dos/ddos mitigation on sdn networks. *IEEE Access*, v. 5, p. 9485–9496, 2017. ISSN 2169-3536. doi: 10.1109/ACCESS.2017.2702341.

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

[37] ZERBINI, C. B. et al. 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.

[38] PENA, E. H. et al. 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.

[39] MACKAY, D. J. C. *Information Theory, Inference & Learning Algorithms*. New York, NY, USA: Cambridge University Press, 2002. ISBN 0521642981.

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

[41] BEN-NAIM, A. *Entropy demystified: the second law of thermodynamics reduced to plain common sense*. World Scientific Publishing Company, 2007. ISBN 9812700552,9789812700551,9812700528,9789812700520,9789812770691. Disponível em: <http://gen.lib.rus.ec/book/index.php?md5=9235720EDFA8467B0F8DFF0AC6DD8921>.

[42] de Assis, M. V. O.; Rodrigues, J. J. P. C.; Proença, M. L. A novel anomaly detection system based on seven-dimensional flow analysis. In: *2013 IEEE Global Communications Conference (GLOBECOM)*. [S.l.: s.n.], 2013. p. 735–740. doi: 10.1109/GLOCOM.2013.6831160.

[43] MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. *Foundations of Machine Learning*. 2. ed. Mit Press, 2018. ISBN 9780262018258. Disponível em: <http://www.jstor.org/stable/j.ctt5hhcw1>.

[44] BURKOV, A. *The Hundred-Page Machine Learning Book*. 1. ed. [S.l.]: Kindle Direct Publishing, 2019. ISBN 9781790485000.

[45] Alpaydin, E. Multilayer perceptrons. In: _____. *Introduction to Machine Learning*. MITP, 2014. Disponível em: <https://ieeexplore.ieee.org/document/6917150>.

[46] MULTILAYER Artificial Neural Network | Simplilearn. <https://www.simplilearn.com/multilayer-artificial-neural-network-tutorial>. Acessado: 2019-01-12.

[47] REDE Perceptron de uma única camada - Embarcado. <https://www.embarcados.com.br/rede-perceptron-de-uma-unica-camada/>. Acessado: 2019-01-12.

[48] Keller, J. M.; Liu, D.; Fogel, D. B. Multilayer neural networks and backpropagation. In: _____. *Fundamentals of Computational Intelligence: Neural Networks, Fuzzy Systems, and Evolutionary Computation*. IEEE, 2016. p. 35–60. Disponível em: <https://ieeexplore.ieee.org/document/7547444>.

[49] BUILDING a deep neural network to improve network accuracy - Neural Networks with Keras Cookbook. <https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781789346640/2/ch02lvl1sec21/building-a-deep-neural-network-to-improve-network-accuracy>. Acessado: 2019-01-12.

[50] OLGAC, A. Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence And Expert Systems*, v. 1, p. 111–122, 02 2011. Disponível em: <https://www.cscjournals.org/library/manuscriptinfo.php?mc=IJAE-26>.

[51] Ramakrishnan, N.; Soni, T. Network traffic prediction using recurrent neural networks. In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. [S.l.: s.n.], 2018. p. 187–193. doi: 10.1109/ICMLA.2018.00035.

[52] Qi, X.; Wang, T.; Liu, J. Comparison of support vector machine and softmax classifiers in computer vision. In: *2017 Second International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*. [S.l.: s.n.], 2017. p. 151–155. doi: 10.1109/ICMCCE.2017.49.

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

[54] WASON, R. Deep learning: Evolution and expansion. *Cognitive Systems Research*, v. 52, p. 701 – 708, 2018. ISSN 1389-0417. doi: 10.1016/j.cogsys.2018.08.023.

[55] LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved. SN -, v. 521, p. 436 EP –, May 2015. Disponível em: <https://doi.org/10.1038/nature14539>.

[56] Gao, X.; Zhang, J.; Wei, Z. Deep learning for sequence pattern recognition. In: *2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)*. [S.l.: s.n.], 2018. p. 1–6. doi: 10.1109/ICNSC.2018.8361281.

[57] SHORT guide on how Deep Learning really works - techburst. <https://techburst.io/short-guide-on-how-deep-learning-really-works-81a588541b24>. Acessado: 2019-01-12.

[58] Albawi, S.; Mohammed, T. A.; Al-Zawi, S. Understanding of a convolutional neural network. In: *2017 International Conference on Engineering and Technology (ICET)*. [S.l.: s.n.], 2017. p. 1–6. doi: 10.1109/ICEngTechnol.2017.8308186.

[59] A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>. Acessado: 2019-01-12.

[60] BALDI, P. Autoencoders, unsupervised learning, and deep architectures. In: GUYON, I. et al. (Ed.). *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*. Bellevue, Washington, USA: PMLR, 2012. (Proceedings of Machine Learning Research, v. 27), p. 37–49. Disponível em: <http://proceedings.mlr.press/v27/baldi12a.html>.

[61] BUILDING an Autoencoder for Generative Models - DEV Community. <https://dev.to/kayis/building-an-autoencoder-for-generative-models-3e1i>. Acessado: 2019-01-12.

[62] UNDERSTANDING LSTM Networks. <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>. Acessado: 2019-01-12.

[63] BUILDING a Recurrent Neural Network - Step by Step - v1. <https://datascience-enthusiast.com/DL/Building_a_Recurrent_Neural_Network-Step_by_Step_v1.html>. Acessado: 2019-01-12.

[64] GRAVES, A. Long short-term memory. In: _____. *Supervised Sequence Labelling with Recurrent Neural Networks*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 37–45. ISBN 978-3-642-24797-2. doi: 10.1007/978-3-642-24797-2_4.

[65] Bengio, Y.; Simard, P.; Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, v. 5, n. 2, p. 157–166, March 1994. ISSN 1045-9227. doi: 10.1109/72.279181.

[66] Bengio, Y.; Frasconi, P.; Simard, P. The problem of learning long-term dependencies in recurrent networks. In: *IEEE International Conference on Neural Networks*. [S.l.: s.n.], 1993. p. 1183–1188 vol.3. doi: 10.1109/ICNN.1993.298725.

[67] PASCANU, R.; MIKOLOV, T.; BENGIO, Y. On the difficulty of training recurrent neural networks. In: *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*. JMLR.org, 2013. (ICML'13), p. III–1310–III–1318. Disponível em: <http://dl.acm.org/citation.cfm?id=3042817.3043083>.

[68] Althubiti, S. A.; Jones, E. M.; Roy, K. Lstm for anomaly-based network intrusion detection. In: *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*. [S.l.: s.n.], 2018. p. 1–3. ISSN 2474-154X. doi: 10.1109/ATNAC.2018.8615300.

[69] HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, v. 9, p. 1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

[70] Vinayakumar, R.; Soman, K. P.; Poornachandran, P. Applying deep learning approaches for network traffic prediction. In: *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. [S.l.: s.n.], 2017. p. 2353–2358. doi: 10.1109/ICACCI.2017.8126198.

[71] BROWNLEE, J. *Long Short-term Memory Networks with Python: Develop Sequence Prediction Models with Deep Learning*. Jason Brownlee, 2017. Disponível em: <https://books.google.com.br/books?id=ONpdswEACAAJ>.

[72] CHO, K. et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. Disponível em: <http://arxiv.org/abs/1406.1078>.

[73] 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.

[74] Qin, G.; Chen, Y.; Lin, Y. Anomaly detection using lstm in ip networks. In: *2018 Sixth International Conference on Advanced Cloud and Big Data (CBD)*. [S.l.: s.n.], 2018. p. 334–337. doi: 10.1109/CBD.2018.00066.

[75] Mirza, A. H.; Cosan, S. Computer network intrusion detection using sequential lstm neural networks autoencoders. In: *2018 26th Signal Processing and Communications Applications Conference (SIU)*. [S.l.: s.n.], 2018. p. 1–4. doi: 10.1109/SIU.2018.8404689.

[76] Lopez-Martin, M. et al. Network traffic classifier with convolutional and recurrent neural networks for internet of things. *IEEE Access*, v. 5, p. 18042–18050, 2017. ISSN 2169-3536. doi: 10.1109/ACCESS.2017.2747560.

[77] KHAN, F. et al. Tsdl: A twostage deep learning model for efficient network intrusion detection. *IEEE Access*, PP, p. 1–1, 02 2019. doi: 10.1109/ACCESS.2019.2899721.

[78] Lin, W. et al. Using convolutional neural networks to network intrusion detection for cyber threats. In: *2018 IEEE International Conference on Applied System Invention (ICASI)*. [S.l.: s.n.], 2018. p. 1107–1110. doi: 10.1109/ICASI.2018.8394474.

[79] NASEER, S. et al. Enhanced network anomaly detection based on deep neural networks. *IEEE Access*, p. 1–1, 08 2018. doi: 10.1109/ACCESS.2018.2863036.

[80] Zhao, G.; Zhang, C.; Zheng, L. Intrusion detection using deep belief network and probabilistic neural network. In: *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*. [S.l.: s.n.], 2017. v. 1, p. 639–642. doi: 10.1109/CSE-EUC.2017.119.

[81] Fu, R.; Zhang, Z.; Li, L. Using lstm and gru neural network methods for traffic flow prediction. In: *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*. [S.l.: s.n.], 2016. p. 324–328. doi: 10.1109/YAC.2016.7804912.

[82] Yuan, X.; Li, C.; Li, X. Deepdefense: Identifying ddos attack via deep learning. In: *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*. [S.l.: s.n.], 2017. p. 1–8. doi: 10.1109/SMARTCOMP.2017.7946998.

[83] Ustebay, S.; Turgut, Z.; Aydin, M. A. Intrusion detection system with recursive feature elimination by using random forest and deep learning classifier. In: *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*. [S.l.: s.n.], 2018. p. 71–76. doi: 10.1109/IBIGDELFT.2018.8625318.

[84] Roy, B.; Cheung, H. A deep learning approach for intrusion detection in internet of things using bi-directional long short-term memory recurrent neural network. In: *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*. [S.l.: s.n.], 2018. p. 1–6. doi: 10.1109/ATNAC.2018.8615294.

[85] Chen, Z. et al. Autoencoder-based network anomaly detection. In: *2018 Wireless Telecommunications Symposium (WTS)*. [S.l.: s.n.], 2018. p. 1–5. doi: 10.1109/WTS.2018.8363930.

[86] Al-Qatf, M. et al. Deep learning approach combining sparse autoencoder with svm for network intrusion detection. *IEEE Access*, v. 6, p. 52843–52856, 2018. doi: 10.1109/ACCESS.2018.2869577.

[87] GENERAL Python FAQ — Python 3.7.4 documentation. <https://docs.python.org/3/faq/general.html#what-is-python>. Acessado: 2019-07-20.

[88] Nagpal, A.; Gabrani, G. Python for data analytics, scientific and technical applications. In: *2019 Amity International Conference on Artificial Intelligence (AICAI)*. [S.l.: s.n.], 2019. p. 140–145. doi: 10.1109/AICAI.2019.8701341.

[89] HOME - Keras Documentation. <https://keras.io/>. Acessado: 2019-07-20.

[90] De Assis, M. V. O. et al. Fast defense system against attacks in software defined networks. *IEEE Access*, v. 6, p. 69620–69639, 2018. doi: 10.1109/ACCESS.2018.2878576.

[91] TF.KERAS.METRICS.ACCURACY | TensorFlow Core r2.0. <https://www.tensorflow.org/api_docs/python/tf/keras/metrics/Accuracy>. Acessado: 2019-01-12.

[92] POLI, A.; CIRILLO, M. On the use of the normalized mean square error in evaluating dispersion model performance. *Atmospheric Environment. Part A. General Topics*, v. 27, p. 2427–2434, 10 1993. doi: 10.1016/0960-1686(93)90410-Z.

[93] BARBETTA, P. A.; BORNIA, A. C.; REIS, M. M. *Estatística para cursos de engenharia e informática*. [S.l.]: Atlas, 2010.