Universidade Estadual de Londrina

DANIEL MATHEUS BRANDÃO LENT

DETECÇÃO DE ANOMALIAS UTILIZANDO REDES NEURAIS PROFUNDAS CONVOLUCIONAIS

LONDRINA

2021

DANIEL MATHEUS BRANDÃO LENT

DETECÇÃO DE ANOMALIAS UTILIZANDO REDES NEURAIS PROFUNDAS CONVOLUCIONAIS

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

2021

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

Sobrenome, Nome.
Título do Trabalho : Subtitulo do Trabalho / Nome Sobrenome. - Londrina, 2017.
100 f. : il.

Orientador: Nome do Orientador Sobrenome do Orientador.
Coorientador: Nome Coorientador Sobrenome Coorientador.
Dissertação (Mestrado em Ciência da Computação) - Universidade Estadual de Londrina, Centro de Ciências Exatas, Programa de Pós-Graduação em Ciência da Computação, 2017.
Inclui bibliografia.

1. Assunto 1 - Tese. 2. Assunto 2 - Tese. 3. Assunto 3 - Tese. 4. Assunto 4 - Tese. I. Sobrenome do Orientador, Nome do Orientador. II. Sobrenome Coorientador, Nome Coorientador. III. Universidade Estadual de Londrina. Centro de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação. IV. Título.

DANIEL MATHEUS BRANDÃO LENT

DETECÇÃO DE ANOMALIAS UTILIZANDO REDES NEURAIS PROFUNDAS CONVOLUCIONAIS

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 – UEL

Msc. Matheus Pereira de Novaes

Londrina, 8 de Fevereiro de 2021.

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

AGRADECIMENTOS

Agradeço principalmente aos meus pais, que sempre valorizaram minha educação e me proporcionaram todas as condições para minha formação. Em especial, agradeço à minha mãe por todo o apoio em momentos complicados. Agradeço à minha irmã e irmão, tios e tias, primos e primas, por me motivarem a melhorar como pessoa e me apoiarem durante minha formação.

Agradeço aos colegas de curso, em especial aos amigos Vitor, Matheus, Vinícius e Henrique, que me deram suporte durante as dificuldades, me ensinaram tantas coisas e me influenciaram a ser quem sou atualmente.

Agradeço também a todos os professores, que tiveram papel fundamental na minha formação não só acadêmica, mas também da pessoa que sou. Em especial, agradeço ao professor Sakuray, que continuou me ensinando mesmo fora da sala de aula; ao professor Evandro, por sua ótima metodologia que fez evoluir como aluno e programador.

Agradeço especialmente ao professor Mario Lemes Proença Jr., pelo convite que deu origem a esse trabalho e toda a sua paciência enquanto trabalhávamos juntos. Agradeço também ao doutorando Matheus Novaes, membro do grupo de pesquisa Orion, por me ajudar em diversos momentos.

Agradeço à Universidade Estadual de Londrina por proporcionar a estrutura e o ensino por meio do curso de ciência da computação. Por fim, agradeço ao CNPq pela bolsa de iniciação científica que foi concedida por meio do Programa de Iniciação Científica PROIC da UEL.

"O temor do Senhor ensina a sabedoria, e
a humildade antecede a honra."
(Bíblia Sagrada, Provérbios 15:33)

LENT, D. M.. **Detecção de Anomalias Utilizando Redes Neurais Profundas Convolucionais**. 2021. 58f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2021.

RESUMO

A Internet é cada vez mais essencial e acessível para as pessoas e possui uma infinidade de aplicações. Em função disso, sistemas em redes se tornam cada vez maiores e mais complexos de se gerenciar, tornando comum a utilização de sistemas de detecção de intrusão. Nesse trabalho, foram desenvolvidas duas abordagens de um sistema de detecção de anomalias que utilizam redes neurais convolucionais profundas em um ambiente de redes definidas por *software*. A primeira busca criar uma assinatura da rede para detectar qualquer tipo de anomalia enquanto a segunda é treinada para detectar ataques distribuídos de negação de serviço e *portscan*. O primeiro modelo não alcançou resultados satisfatórios, já o segundo consegue detectar os ataques com mais de 97% de precisão. Concluiu-se que o primeiro modelo pode ser melhorado, enquanto o segundo apresenta resultados satisfatórios contra os ataques utilizados no treinamento.

Palavras-chave: Aprendizagem de Máquina; Detecção de Anomalia; Redes Definidas por *Software*; Redes Neurais Convolucionais.

LENT, D. M.. **Anomaly Detection Using Convolutional Neural Networks**. 2021. 58p. Final Project (Bachelor of Science in Computer Science) – State University of Londrina, Londrina, 2021.

ABSTRACT

More and more, Internet is essential and acessible for people and has endless possible applications. As a result, network systems become larger and more complex to manage, making common the adoption of intrusion detection systems. In this work, two approaches of anomaly detection systems that use deep convolutional neural networks in a software defined network environment were developed. The first aims to create a signature that represents the network normal behavior to detect any type of anomaly, while the other is trained to detect distributed denial of service and portscan attacks. The first model didn't achieve acceptable results whilst the second can detect attacks with more than 97% precision. It follows that the first model can be refined and the second presents solid results aganist the attacks used on training.

Keywords: Machine Learning. Convolutional Neural Networks. Anomaly Detection

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AMD	*Advanced Micro Devices*
CNN	*Convolutional Neural Network*
DDoS	*Distributed Denial of Service*
DoS	*Denial of Service*
FTP	*File Transfer Protocol*
GAN	*Generative Adversarial Network*
GB	*Gigabyte*
IMDB	*Internet Movie Data Base*
IoT	*Internet of Things*
IP	*Internet Protocol*
LFW	*Labeled Faces in the Wild*
LSTM	*Long Short-Term Memory*
ReLU	*Rectified Linear Unit*
RNN	*Recurrent Neural Network*
RTX	*Ray Tracing Texel eXtreme*
SDN	*Software Defined Networks*
UEL	*Universidade Estadual de Londrina*
URL	*Uniform Resource Locator*

LISTA DE SÍMBOLOS

Σ	Letra grega Sigma, maiúscula, indica somatório
σ	Letra grega sigma, minúscula, se refere à função de ativação *sigmoid*
$tanh()$	Função tangente hiperbólica
$relu()$	Função *Rectified Linear Unit*
h	Horas
H	Entropia
p_i	Probabilidade de i
O	Dimensão resultante da convolução
e	Número de Euler

SUMÁRIO

1 INTRODUÇÃO

A *Internet* é cada vez mais essencial para o cotidiano das pessoas devido a sua infinidade de possíveis aplicações. Dentre os serviços oferecidos por meio dela estão comunicação em tempo real, compras, armazenamento e processamento de dados, transmissão de conteúdo, além de outros não citados [1]. Além disso, com a criação de novos serviços, são necessários cada vez mais recursos e estruturas cada vez mais complexas para sustentá-los e administrá-los [2].

A grande quantidade de ativos de rede de diversos fabricantes, bem como os diversos recursos que cada um apresenta, impõem um aumento na complexidade para gestão destes recursos. Isto tudo motivou na criação de um paradigma chamado Redes Definidas por *Software*(*SDN*, do inglês *Software Defined Networking*), que tem como objetivo simplificar o gerenciamento de uma rede ao separar o plano de controle do plano de dados, possibilitando uma modularização da rede e um controle mais simples a partir de protocolos que servem como interface para os sistemas de segurança administrarem cada dispositivo, além de outras aplicações [3][4]

Neste contexto, verifica-se que servidores de serviços de rede podem ficar indisponíveis por motivos como: problemas de *hardware*, falta de energia, problemas com *software* ou até, e foco desse trabalho, ataques externos maliciosos originados da *Internet*. Tais ataques são chamados de anomalias e podem ocorrer de diversas formas com diferentes consequências. Por exemplo, um volume de acessos a um site que está mais alto que o normal devido a atacantes, sobrecarregando o servidor e impossibilitando novas conexões, ou até uma tentativa de quebra de senha por força bruta são ataques que precisam ser detectados e mitigados antes que causem algum dano.

Um dos ataques de rede mais recorrentes nos últimos anos é o ataque de negação de serviço (*DoS*, do inglês *Denial of Service*) [5]. Esse tipo de ataque sobrecarrega o servidor com requisições de agentes maliciosos impossibilitando que usuários legítimos consigam acesso à aplicação [1][6][7]. O ataque ainda pode vir de forma distribuída (*DDoS, do inglês Distributed Denial of Service*), o que dificulta ainda mais a identificação dos usuários regulares dentre os agentes maliciosos [8]. Nesse formato, o atacante utiliza diversos dispositivos conectados à *Internet* de diferentes tipos e locais para atacarem ao mesmo tempo.

Outro ataque importante estudado nesse trabalho é o de *portscan*. Nele, o agente malicioso procura coletar informações sobre portas abertas em um sistema para se preparar para um ataque [9]. Nesse ataque, pacotes são enviados a diferentes portas do *host* alvo. Assim, a porta é identificada como aberta ou fechada dependendo da resposta recebida

pelo invasor [10].

Com a facilidade de acesso à *Internet* e a popularização de dispositivos de *Internet das Coisas* (*IoT*), como roteadores, geladeiras e televisões inteligentes, câmeras conectadas à *Internet*, assistentes pessoais, etc. Tornou-se mais comum para agentes maliciosos encontrarem equipamentos que possam infectar com *botnets* e utilizar para um ataque distribuído de negação de serviço, devido à alta vulnerabilidade desses dispositivos [11][12]. Os equipamentos infectados continuam a operar normalmente enquanto aguardam o comando para atacar a vítima, por isso o *malware* não é percebido pelo dono do aparelho, enquanto o agressor continua em busca de novas unidades.

Esse tipo de problema é combatido pelos sistemas de detecção de intrusão do servidor atacado, que identificam o que está ocorrendo para que medidas sejam tomadas, seja pelo administrador de rede ou pelo próprio sistema [13]. Esse tipo de sistema precisa ser capaz de identificar atividade anômala sem prejudicar o desempenho da aplicação que está protegendo [14]. A identificação é feita ao analisar dados sobre a rede em uma determinada frequência para decidir se naquele momento existe tráfego considerado anômalo [13]. Contudo, a determinação do que é considerada uma anomalia para diferentes redes pode ser vaga e depende de diversos fatores, tornando o assunto amplamente estudado [7][13][15][16].

O ramo de Aprendizagem de Máquina é estudado pela possibilidade de resolver problemas utilizando experiência e estatística ao invés de programação explícita [17]. Particularmente, redes neurais são uma abordagem que apresentam alto desempenho quando o problema envolve de reconhecimento de padrões [18] e já foram diversas vezes utilizadas para detecção de anomalias em redes [19][7].

Existem diferentes tipo de redes neurais, um deles é chamado de convolucional. Nesse tipo de rede, o processo de convolução é o que as define. Esse processo extrai características importantes sobre os dados ao mesmo tempo que os simplifica [20]. Normalmente, esse tipo de rede tem como foco o processamento de imagens [21], mas já foram aplicadas para detecção de ataques distribuídos de negação de serviço [22].

Neste trabalho, foi estudada a aplicação de redes neurais convolucionais para detecção de anomalias em redes. Para isso, foi realizado um estudo de caso utilizando dados que emulam uma rede definida por *software* que recebeu ataques *DDoS* e *portscan* em determinados horários do dia. O objetivo é utilizar a capacidade de extração de características e reconhecimento de padrões das convoluções [23] para reconhecer momentos em que há ataques à rede durante um dia qualquer, coletando e analisando o tráfego de rede a cada segundo. O desempenho será verificado principalmente pela taxa de verdadeiros positivos e negativos em situações com tráfego tanto normal quanto anômalo. A partir da detecção dos ataques, ações de mitigação podem ser tomadas e o gerente de rede pode ser alertado para analisar o problema e futuramente mitigá-lo.

O restante do trabalho está organizado da seguinte maneira: o Capítulo 2 descreve conceitos fundamentais para realização do estudo tais como, detecção de intrusão e anomalias, entropia e ataques de negação de serviço. Também é brevemente apresentado o conceito de aprendizagem de máquina, redes neurais e algumas metodologias de redes neurais profundas e aplicações. O Capítulo 3 apresenta conceitos sobre redes neurais convolucionais, em que é descrito como funciona a convolução, o aprendizado e aplicações em diferentes áreas. O Capítulo 4 apresenta alguns trabalhos relacionados ao tema. O capítulo 5 contém um estudo de caso, no qual é desenvolvido um detector de ataques distribuídos de negação de serviço e portscan utilizando redes neurais convolucionais a partir de um conjunto de dados de ataques em um cenário de rede emulado. Por fim, no Capítulo 6, apresenta conclusões tomadas e possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Redes Definidas por Software

Com o desenvolvimento de tecnologias e aumento na quantidade de ativos de rede necessários para sua aplicação, é esperada uma maior complexidade para gerenciar o funcionamento dos equipamentos, necessitando de cada vez mais recursos e tempo [3]. Para resolver esse problema, um novo paradigma de organizar a rede foi desenvolvido. O conceito de Redes Definidas por *Software* (*SDN*) é um paradigma que busca simplificar o gerenciamento de uma rede ao separar o plano de controle do plano de dados, criando uma modularização da rede e aumentando sua programabilidade [4].

Dessa forma, o plano de dados é encarregado do encaminhamento de pacotes, comunicação entre dispositivos, além de conter ferramentas de inspeção de pacotes [24]. Esse plano, entretanto, não decide por si só como é feito o encaminhamento de pacotes e gerenciamento do tráfego; as "instruções" vêm do plano de controle, que atua como um intermediador entre os ativos de rede e as aplicações que os gerenciam e os monitoram [24]. A partir do plano de controle, é centralizado o gerenciamento de cada *switch* da rede, simplificando a coleta e envio de informações para cada dispositivo [3].

Para possibilitar a comunicação entre o plano de controle e de dados, é necessária a determinação de uma interface chamada *Southbound Interface*. Um dos protocolos comumente utilizados como interface é o protocolo *OpenFlow*. Esse protocolo viabiliza a comunicação do plano de controle com cada *switch* e a programação do controle de fluxo a partir de uma tabela de fluxo (*Flow Table*), que será utilizada por cada *switch* para decidir sobre como tratar cada pacote recebido [3]. Além disso, ele também possibilita a coleta de dados de fluxo de cada dispositivo, recurso utilizado na detecção de anomalias em redes definidas por software [25].

Já a comunicação entre o plano de controle e as aplicações da rede é proporcionada pela *Northbound Interface*. Normalmente, essas interfaces são responsáveis por possibilitar requisições e a visualização da rede e seu comportamento [24].

2.2 Detecção de Intrusão

Detecção de intrusão em redes se torna cada vez mais essencial à medida que a *Internet* se torna mais presente no cotidiano. A segurança e estabilidade das aplicações são fatores imprescindíveis para sua viabilidade e sucesso. Conforme a acessibilidade da *Internet* aumenta, o volume de dados a serem analisados também cresce, ao mesmo tempo que novos tipos de ataque são criados, tornando cada vez mais complexo o desenvolvimento

de defesas efetivas [26][27]. Por isso, sistemas de detecção de intrusão são estudados e estão em constante evolução.

Sistemas de detecção de intrusão são uma combinação de *software* e *hardware* para monitorar atividades na rede e detectar ações maliciosas [13]. Existem diversas técnicas que podem ser aplicadas para solucionar o problema, cada uma com pontos fortes e fracos. Para avaliá-las, não só fatores como taxa de detecção correta podem ser utilizados, mas também a taxa de alarmes falsos, seus custos, confiabilidade contra ataques novos e uso de processamento e memória devem ser considerados [13][26][28].

Sistemas de detecção de intrusão podem ser classificados pela plataforma que coleta os dados a serem analisados, sendo eles baseado em rede (*Network Based*), baseado em *host* (*Host Based*) e híbrido [2]. Aqueles baseados em rede têm como objetivo proteger vários pontos de conexão coletando dados de tráfego de cada um. Esse tipo de sistema pode analisar completamente os pacotes que passam pelos sensores, incluindo endereços de *IP* (do Inglês *Internet Protocol*), porta e conteúdo. Contudo, pode se tornar inviável analisar todos os pacotes que percorrem determinado ponto quando o tráfego é elevado [2].

Já os sistemas baseados em *host* têm como objetivo proteger apenas o computador monitorado. É possível detectar tanto atividades maliciosas contra aquele *host* quanto comportamento suspeito originado de usuários da máquina [2]. Por analisar somente um único computador, esse sistema pode utilizar dados mais específicos para avaliar sua segurança, como monitorar chamadas e *logs* de sistema e abuso de privilégios [2].

Por último, sistemas híbridos buscam combinar os tipos anteriores para se aproveitar dos benefícios de cada um. Contudo, são complexos de implementar por precisar de dados vindo tanto dos sensores da rede quanto dos *hosts*, necessitando de técnicas que combinem as duas formas de informação [2].

Um tipo de técnica de detecção de intrusão é a baseada em anomalia, que analisa o fluxo da rede. É chamado de fluxo um conjunto de pacotes que passam por algum ponto da rede em determinado intervalo de tempo [29]. O sistema coleta os pacotes, adiciona rótulos como horário de recebimento, retira seu conteúdo e os armazena em um banco de dados. A partir desse banco, os dados são analisados e pré-processados para que características sejam extraídas e enviadas a uma máquina de decisão, que irá alertar caso detecte aquele fluxo como um ataque [29].

Esse tipo de técnica, por utilizar principalmente dados de cabeçalho de pacotes sem acessar seu conteúdo, tem como vantagem a possibilidade de analisar volumes de tráfego maiores do que técnicas que analisam o pacote completo e pode ser utilizada em aplicações em que seu conteúdo é criptografado. Além disso, os pontos de coleta de fluxo podem ser espalhados pela rede e ter seus dados agregados ao banco de dados. Entretanto,

esse tipo de técnica não é tão precisa quanto as que analisam o conteúdo dos pacotes [29].

Um outro tipo de técnica é a baseada em assinatura, que compara o conteúdo de pacotes recebidos com padrões encontrados em dados de intrusões previamente analisadas e identificadas [30][31]. Uma de suas vantagens é a elevada precisão ao detectar os ataques que já são conhecidos e algumas de suas variações, enquanto tem como maior problema a dificuldade em detectar anomalias desconhecidas, já que ainda não possuem uma entrada no conjunto de padrões a serem reconhecidos [26][32].

2.3 Detecção de Anomalias em Redes

Detecção de anomalias é outro tipo de detecção de intrusão. Uma anomalia é um comportamento incomum, diferente do comportamento normal ou esperado dos ativos de rede [2]. Sabendo disso, é possível perceber uma irregularidade analisando o comportamento em tempo real desses ativos, enquanto se conhece o que é normal para cada momento. Entretanto, determinar o que é normal em uma rede pode ser uma tarefa não trivial, já que existem diversos parâmetros que podem ser medidos e derivados para encontrar uma definição, como a quantidade e tamanho de pacotes, entropia de endereços de *IP* de origem e destino, dentre outros [15][14][33][34].

Esse tipo de método de detecção é o baseado em anomalia (*Anomaly Based*) e pode ser aplicado utilizando modelos estatísticos [2]. São coletados dados sobre o fluxo da rede para que seja feita uma previsão sobre o comportamento da rede em determinado momento e então é comparado com o tráfego normal da rede [15][34]. A partir disso, é possível verificar se há anomalias, como um endereço que se repete excessivamente, um número de requisições maior que o normal, dentre outros [25][35].

Na literatura pode ser encontrado trabalhos que aplicam essa metodologia. Dentre eles, pode ser citado (Hamamoto et al. 2018), que utiliza algoritmo genético para prever o comportamento da rede e lógica difusa para decidir se há ou não anomalia [1], (Carvalho et al. 2016) e (Assis et al. 2013) buscam melhorar a eficiência da detecção utilizando uma modificação da heurística de colônia de formigas [36][37], (Proença et al. 2004) que gera uma assinatura digital para caracterizar a rede [34], e (Novaes et al. 2020) que utiliza aprendizado profundo para prever o comportamento da rede [38].

Fernandes et al. [2] indicam duas formas de categorização de anomalias: baseadas em sua natureza e baseadas em seu agente causador, a primeira não leva em consideração a existência de caráter malicioso, diferentemente da segunda. As categorias por natureza são: pontual, quando a anomalia acontece de forma isolada; coletiva, quando um conjunto de eventos isoladamente é considerado normal, porém quando ocorrem em uma sequência específica é considerada uma anomalia; e contextual, quando um evento é considerado anômalo devido a informações adicionais que o acompanham, por exemplo,

tempo, contexto geográfico, etc.

As categorias por agente causador são divididas em 4 classes: operacional, que geralmente é causada por erros de configuração, falha de *hardware*, etc; *flash crowds*, que são causadas por uma elevada quantidade de usuários legítimos tentando acessar um recurso; anomalias de medição, que acontecem quando ocorre um erro durante a coleta de dados utilizados para caracterizar o tráfego; e por fim, os ataques maliciosos, que têm como objetivo negar um serviço ou atrapalhar o sistema [2].

2.3.1 Ataques de Negação de Serviço

Ataques de negação de serviço (*Denial of Service* - DoS) são um tipo de anomalia que utiliza um grande volume de requisições para sobrecarregar uma vítima, que pode ser um servidor ou até uma rede [39][1]. Com a sobrecarga, a vítima não consegue atender todas as requisições, impedindo seu funcionamento e inviabilizando o uso por usuários legítimos. Nesse tipo de ataque, o volume de requisições necessárias para incapacitar o atacado deve ser maior que sua capacidade de processá-las. Por isso, também é comum que as investidas venham de forma distribuída, o que torna possível alcançar uma maior carga de dados. Os ataques distribuídos recebem o nome *Distributed Denial of Service* (DDoS), e são ainda mais problemáticos, já que vêm de diversas origens ao mesmo tempo, impossibilitando a identificação de usuários maliciosos entre os legítimos [8][1].

Para realizar um ataque distribuído, são necessários vários dispositivos que possam fazer requisições ao mesmo tempo. Para isso, os atacantes invadem dispositivos que fiquem constantemente ligados, conectados à *Internet* e com baixa segurança contra *malwares*; características essas que dão liberdade ao atacante de preparar o ataque com antecedência e dispará-lo no momento mais apropriado. O que facilita esse tipo de prática é o crescimento da quantidade de dispositivos de *Internet das Coisas* (IoT) presentes em cada casa; babá eletrônica, geladeira inteligente, televisões, câmeras, cafeteiras e roteadores são exemplos de aparelhos que podem ter problemas de segurança e não são notados pelos moradores da casa [11].

Para infectar os dispositivos, *botnets* como Mirai [12] escaneiam endereços de *IP* públicos em portas que normalmente equipamentos de IoT utilizam. O *bot* faz testes de força bruta para descobrir suas senhas. Depois de invadir o aparelho, o programa envia suas características de volta para o servidor utilizando uma porta diferente. Depois de decidir quais dispositivos infectar, o invasor envia um comando para que o aparelho faça o download e execute o *malware*, que por si só começa a proteger a si mesmo de outros *malwares* fechando pontos de possíveis invasões. Nesse ponto, o equipamento está aguardando pronto para atacar ao comando do controlador [12]. Na Figura 1, uma representação de ataque *DDoS*.

Uma das características dos ataques *DDoS* é que os *IPs* maliciosos tendem a se

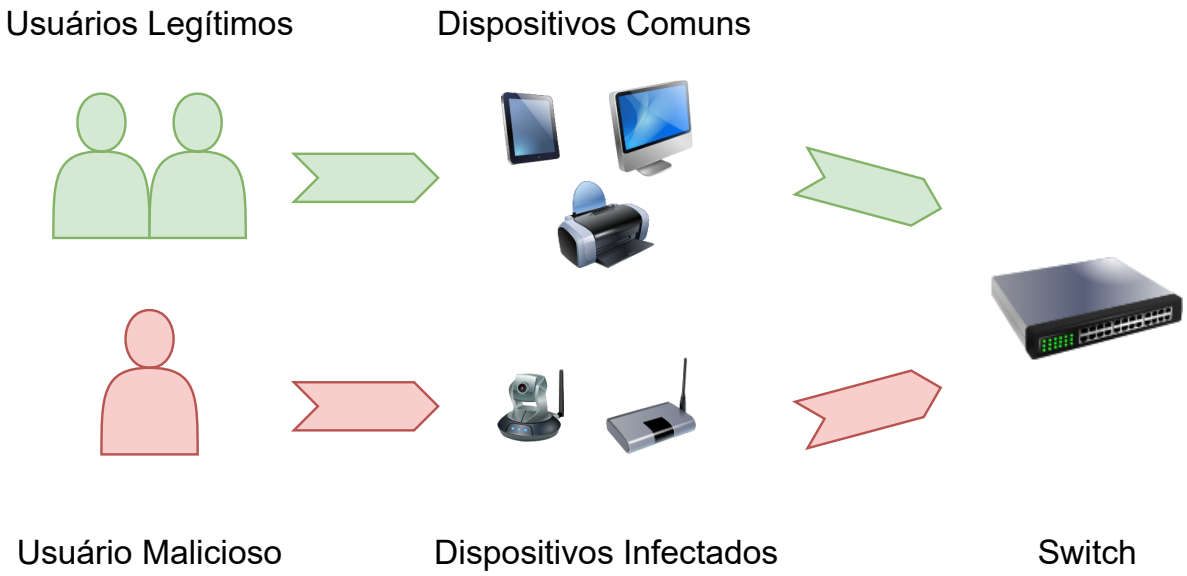

Figura 1 – Representação de ataque *DDoS*. Fonte: O próprio autor.

repetir a cada requisição, o que implica na diminuição da entropia de endereços *IP*. Por isso, a entropia de Shannon pode ser útil na detecção dos ataques enquanto estão sendo iniciados. Assim, medidas de mitigação podem ser tomadas antes que o serviço deixe de funcionar ou, ao menos, minimizando o período de instabilidade [1][8].

Existem diferentes tipos de ataques *DDoS* e para cada um algumas técnicas de mitigação podem ser utilizadas. Rai e Challa [6] ilustram diversas abordagens enquanto pontuam algumas de suas limitações. Um dos métodos mencionados [40] propõe um sistema de defesa para *websites* que analisa informações como endereço de *IP*, identificador do usuário, seu nível de acesso permitido, a *url* (*Uniform Resource Locator*) de destino e o serviço requisitado de cada acesso. Ao encontrar discrepâncias entre as informações, como a tentativa de acesso a uma página sem permissão, um ataque é identificado e o *IP* de origem é bloqueado. Ao mesmo tempo, um modelo estatístico de Markov estabelece limiares sobre informações de comportamento de um usuário, como quantidade média de páginas acessadas e o número de requisições, para classificar como normal ou malicioso.

2.3.2 Ataques *Portscan*

Ataques portscan são uma forma de coletar informações sobre uma vítima para se preparar para um futuro ataque [9]. Normalmente, as informações coletadas são uma lista de portas abertas, em que requisições podem ser feitas ao *host*. Para realizar o ataque, pacotes são enviados a diversas portas do alvo, assim, dependendo da resposta recebida, a porta é identificada como aberta ou fechada [10].

2.4 Aprendizagem de máquina

Aprendizagem de máquina (do Inglês Machine Learning) é o ramo da computação que desenvolve algoritmos que dão a computadores a habilidade de aprender sem serem explicitamente programados [41]. A tarefa pode variar de diversas formas, algumas delas são, por exemplo, uma classificação, como detectar se um e-mail é um *spam*[17]; pode ser uma previsão, como prever suscetibilidade a câncer [42] ou até jogar jogos como xadrez [43]. A experiência vem de dados de entrada que o computador deve utilizar para encontrar padrões. Tais dados devem ser uma representação do objeto estudado. Por exemplo, se o objeto de estudo é uma pessoa, os valores sobre ela podem assumir a forma de um vetor, em que cada posição representa uma característica, como altura, idade, massa etc. Essas características são chamadas de "*features*". Com informações e treino o suficiente, é possível predizer informações sobre uma pessoa nunca antes conhecida [17][44].

Dentro do ramo de *machine learning*, existem tipos diferentes de algoritmo. O mais comum deles é o de aprendizagem supervisionada. Nesse método, os dados vêm acompanhados de rótulos, que são os valores que devem ser retornados. Dessa forma, em um exemplo simples, o computador coloca os dados em um plano de N dimensões, em que N é o número de "*features*" da entrada e traça um hiperplano chamado de limite de decisão que divide os grupos de diferentes rótulos. Assim, ao encontrar casos desconhecidos, basta o algoritmo colocar a entrada no plano e comparar com sua posição em relação ao limite de decisão para atribuir um rótulo a ele. Para encontrar esse limite, o algoritmo utiliza uma função de otimização que, em geral, busca minimizar o erro [17][44].

Contudo, às vezes não é possível ter uma base de dados rotulada, seja porque são escassos ou custosos. Nesse caso é comum utilizar-se de técnicas de aprendizagem não supervisionada, que consistem em transformar o que está sendo recebido em informações que possam ser utilizadas para resolver o problema. Um exemplo é a técnica de agrupamento, que cria seus próprios rótulos colocando a entrada em um plano e atribuindo uma categoria a cada grupo de itens mais próximos. Outro exemplo é a técnica de detecção de anomalia, onde o algoritmo atribui um valor a cada dado, baseado no quanto ele é diferente dos demais [17].

2.4.1 Regressão

Outro tipo de algoritmo é aquele que utiliza regressão. Nesse modelo, uma variável resultante y é calculada a partir de um vetor de variáveis de entrada $\overrightarrow{x}$. Para calcular o resultado corretamente, é necessário um conjunto de dados já rotulados que serão utilizados para encontrar uma função que representa o objeto estudado. Tal função tem como entrada o vetor $\overrightarrow{x}$ e saída o resultado esperado y. Para exemplificar, um tipo simples de regressão é a linear.

Na regressão linear, o vetor $\overrightarrow{x}$ tem tamanho 1, dessa forma, é possível encontrar uma função do tipo $ax + b = y$ para solucionar o problema. Para encontrar a função, é feito o seguinte cálculo: inicia-se uma matriz A com n linhas e duas colunas, onde n é o número de pontos de entrada. Sua primeira coluna será composta pelas variáveis independentes de cada ponto da entrada; já a segunda é preenchida com 1s. O vetor $\overrightarrow{m}$, que deseja-se encontrar terá duas linhas. Por fim, o vetor y terá n linhas e conterá as variáveis resultantes dos pontos de entrada. A fórmula está representada na Equação 2.1

$$A = \begin{bmatrix} x_1 & 1 \\ x_2 & 2 \\ x_3 & 3 \\ . & . \\ x_n & n \end{bmatrix}, \overrightarrow{m} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}, \overrightarrow{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ y_n \end{bmatrix} \tag{2.1}$$

O vetor m deve satisfazer a seguinte equação:

$$A\overrightarrow{m} = \overrightarrow{y}$$

Para isso, o cálculo na Equação 2.2 é feito:

$$\begin{aligned} A\overrightarrow{m} &= \overrightarrow{y} \\ A^T A\overrightarrow{m} &= A^T\overrightarrow{y} \\ (A^T A)^{-1}A^T A\overrightarrow{m} &= (A^T A)^{-1}A^T\overrightarrow{y} \\ I\overrightarrow{m} &= (A^T A)^{-1}A^T\overrightarrow{y} \\ \overrightarrow{m} &= (A^T A)^{-1}A^T\overrightarrow{y} \end{aligned} \tag{2.2}$$

Para um aprendizado eficiente, é importante não só ter uma quantidade de dados relevante, mas também assegurar sua qualidade. Por isso, além de possuir uma amostra significativa de cada tipo de exemplo, os rótulos devem estar corretos. A dificuldade da tarefa pode aumentar consideravelmente caso as amostras possuam erros, já que são cruciais para o aprendizado [44].

2.5 Redes Neurais

Uma rede neural é um sistema que, inspirado no cérebro humano, é composto por neurônios que possuem conexões entre si. Os neurônios e suas conexões são organizados em camadas, de forma que os dados recebidos começam na camada de entrada e são processados de camada em camada até chegar na camada de saída, produzindo um resultado

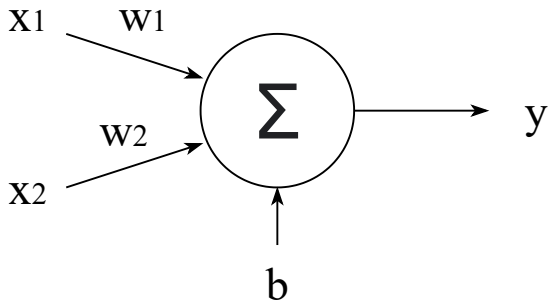

Figura 2 – Um *perceptron*. Fonte: O próprio autor.

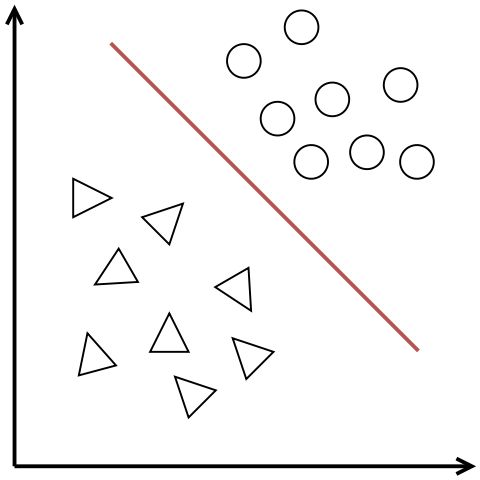

Figura 3 – Problema Linearmente Separável. Fonte: O próprio autor.

em relação ao dado de entrada. Nessa rede são aplicados processos de aprendizado para que ela alcance seu objetivo [18].

Um neurônio, que também pode ser chamado de *perceptron*, é a unidade básica de processamento da rede neural representada na Figura 2. Ele é composto por um conjunto de ligações chamadas sinapses, cada uma com um valor de peso relacionado (na figura representado como W, do ingês *weight*); um somador, que calcula a soma das ligações sinápticas junto de seus respectivos pesos, um viés (Na figura representado por b, do inglês *bias*), que modifica o resultado da soma anterior; e por fim, uma função de ativação, que limita as possibilidades de saída do *perceptron* de acordo com seu objetivo [45].

Dessa forma, um neurônio sozinho pode servir como um classificador binário para problemas em que uma única linha de divisão em um plano pode separar duas classes distintas corretamente. É possível observar na Figura 3 um exemplo desse tipo de problema em que uma reta que passa pelo segmento vermelho consegue separar as classes Círculo e Triângulo perfeitamente [45].

O aprendizado de cada neurônio vem do ajuste do peso das conexões sinápticas e seu viés. A diferença entre o valor esperado e a resposta do neurônio ao aplicar uma

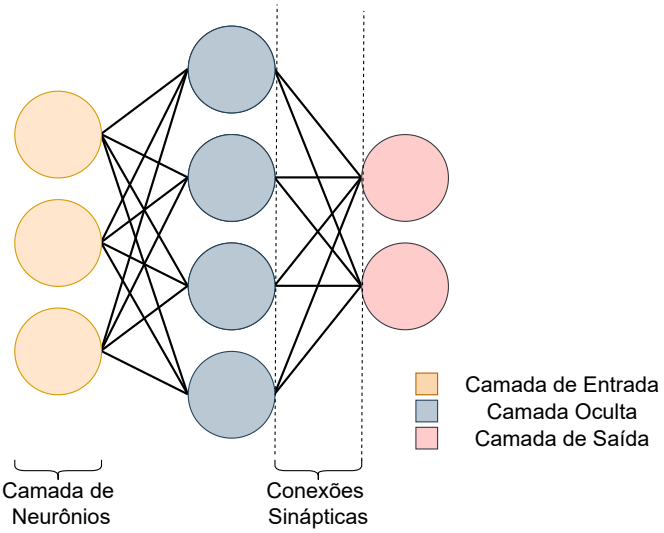

Figura 4 – Rede Neural Multicamada. Fonte: O próprio autor.

entrada é o chamado erro. Assim, os pesos e viés são ajustados em função do valor de entrada e do erro calculado [18][45].

Em problemas mais complexos, um único *perceptron* não é suficiente para atingir o resultado esperado. Por isso, são combinados vários *perceptrons* para conseguir uma solução. Em geral redes neurais possuem neurônios com funções de ativação diferentes, que formam camadas que são conectadas entre si por conexões sinápticas com pesos diferentes. As camadas são divididas em camadas de entrada, camadas ocultas e saída, como ilustrado na Figura 4. [45]

2.5.1 Funções de ativação

As funções de ativação têm como finalidade limitar o intervalo de saída de um neurônio e, de certa forma, decidir o quão relevante é a informação que o neurônio está enviando. Dessa forma, a resposta do neurônio pertencerá ao intervalo da imagem da função de ativação e poderá ser mais ou menos influente dependendo do valor que a função retorna. [46]

2.5.1.1 Sigmoid

A função *sigmoid*, descrita na Equação 2.3 tem como domínio o conjunto dos números reais e imagem valores entre 0 e 1; é continuamente derivável [47]. É comum a utilização de *sigmoid* para problemas que têm como resposta valores que representem a probabilidade ou grau de certeza sobre algo [48].

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

2.5.1.2 Tanh

Tanh é a função da tangente hiperbólica representada na Equação 2.4 e é similar à *Sigmoid*. Seu domínio é o conjunto dos reais e sua imagem varia entre -1 e 1. É continuamente derivável e pode ser escolhida no lugar da *sigmoid* por ter 0 como centro e por ter um gradiente mais inclinado [47].

$$tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{2.4}$$

2.5.1.3 ReLU

ReLu é a abreviação de *Rectified Linear Unit*, que tem como domínio todos os reais e imagem todos os reais positivos e 0. Ela é derivável em todo seu domínio exceto no ponto $x = 0$. Essa função é mais eficiente que outras por não ativar todos os neurônios ao mesmo tempo, ao invés disso, somente alguns estão ativos enquanto outros retornarão 0 [47]. Um dos problemas dessa função é que sua derivada para valores menores que 0 sempre será 0, implicando que o neurônio não aprenderá nada quando alcança esse resultado [47]. Sua equação pode ser representada utilizando a função de máximo entre 0 e x, como na Equação 2.5.

$$relu(x) = max(0, x) \tag{2.5}$$

2.6 Redes Neurais Profundas

A principal diferença entre redes neurais profundas e rasas é a quantidade de camadas de neurônios utilizadas em cada uma. [18] Mais camadas também significa mais neurônios, o que aumenta a quantidade de elementos treináveis da rede neural. Com isso, é possível atingir uma maior capacidade de generalização em comparação a redes neurais mais simples [23][49]. Contudo, a complexidade da rede também aumenta o quanto deve ser treinada para atingir resultados satisfatórios [18][48].

2.6.1 Redes Neurais Recorrentes

Redes recorrentes são caracterizadas por possuírem conexões que levam em consideração respostas anteriores para processar elementos seguintes. Dessa forma, a rede neural pode usar as entradas anteriores como contexto para as próximas. Por isso, esse tipo de rede é utilizada principalmente para problemas que envolvem algum tipo de entrada sequencial e que utilizam contexto para serem solucionados, por exemplo, reconhecimento de fala, composição de música, detecção de anomalias e previsão de séries temporais [38][50][51][52][53] .

Um exemplo de rede neural recorrente (*RNN*) é a rede *Long Short-Term Memory* (*LSTM*), que tem como diferencial a possibilidade de utilizar um contexto mais distante que uma *RNN* comum. A unidade *LSTM* utiliza módulos menores para decidir se alguma informação anterior é relevante para o dado atual, se alguma informação anterior deve ser descartada e o que deve ser adicionado ao contexto para as próximas informações [54][55].

2.6.2 Redes Generativas Adversárias

Redes Generativas Adversárias (*GAN*, do inglês *Generative Adversarial Network*) são um modelo de aprendizagem profunda que utiliza uma competição entre duas redes para melhorar o desempenho de ambas. Sua utilização é comum em problemas de síntese, edição e classificação de imagens. Em geral, o método é aplicado da seguinte forma: uma das redes será a geradora enquanto a outra é a discriminadora. O trabalho da discriminadora é descobrir se o dado que está recebendo é real ou se foi gerado pela outra rede. Já a geradora irá gerar dados que se passem por verdadeiros, enviando para a outra rede e recebendo seu *feedback*. Para o treinamento, algumas técnicas são recomendadas para que a convergência seja possível. Inicialmente, é atualizada a discriminadora somente até conseguir distinguir do gerador atual, em seguida, o gerador é atualizado até que o discriminador não consiga distinguir entre dados reais e falsos [56].

Em [57] são apresentadas algumas aplicações de *GAN*s. Sobre geração de imagens, uma das aplicações citadas é a de reconstrução e pintura. Nesse caso, uma imagem incompleta é inserida no gerador para sua reconstrução. Isso pode ser utilizado não só para restauração de imagens antigas ou corrompidas, mas também para remover objetos indesejados de uma imagem. Outro tipo de aplicação é a de processamento de fala, em que a rede recebe uma imagem de um ser humano e um segmento áudio para gerar um vídeo com sincronização labial e movimentos faciais como olhos piscando e movimento de sobrancelhas.

2.6.3 Redes Autoencoder

O tipo *autoencoder* tem como principal vantagem a possibilidade de treinamento não supervisionado. Seu objetivo é aprender formas de representar os dados da entrada em menores dimensões, descartando informações desnecessárias de forma que ainda seja possível recuperar o máximo possível do original. Para isso, a arquitetura de uma rede neural *autoencoder* é caracterizada por dois aspectos essenciais para seu funcionamento. Primeiro, o número de neurônios da camada de entrada deve ser o mesmo da camada de saída; isso é necessário porque o objetivo é recriar os mesmos dados recebidos pela primeira camada na última. Segundo, é necessário que as camadas ocultas tenham menos neurônios que as de entrada e saída; o processo de diminuição da dimensão dos dados se dá com a rede encontrando formas de utilizar menos neurônios para armazenar as

informações e recuperá-las em seguida. Para o seu treinamento não são necessários rótulos pois os próprios dados de entrada são os objetivos; assim, o erro é calculado pela diferença entre a saída e o dado original [58]. Na Figura 5 é ilustrado um exemplo de arquitetura *autoencoder*.

É possível que, durante o treinamento, a rede neural aprenda simplesmente a multiplicar a entrada por 1, não alterando nem aprendendo nada, enquanto alcança um resultado perfeito. Isso é um problema, pois não há aprendizado sobre o comportamento dos dados, apenas uma transferência de valores da entrada para a saída. Uma das soluções é adicionar ruído à entrada enquanto tem como objetivo de saída os valores originais, forçando a rede a extrair informações de dados imperfeitos [58].

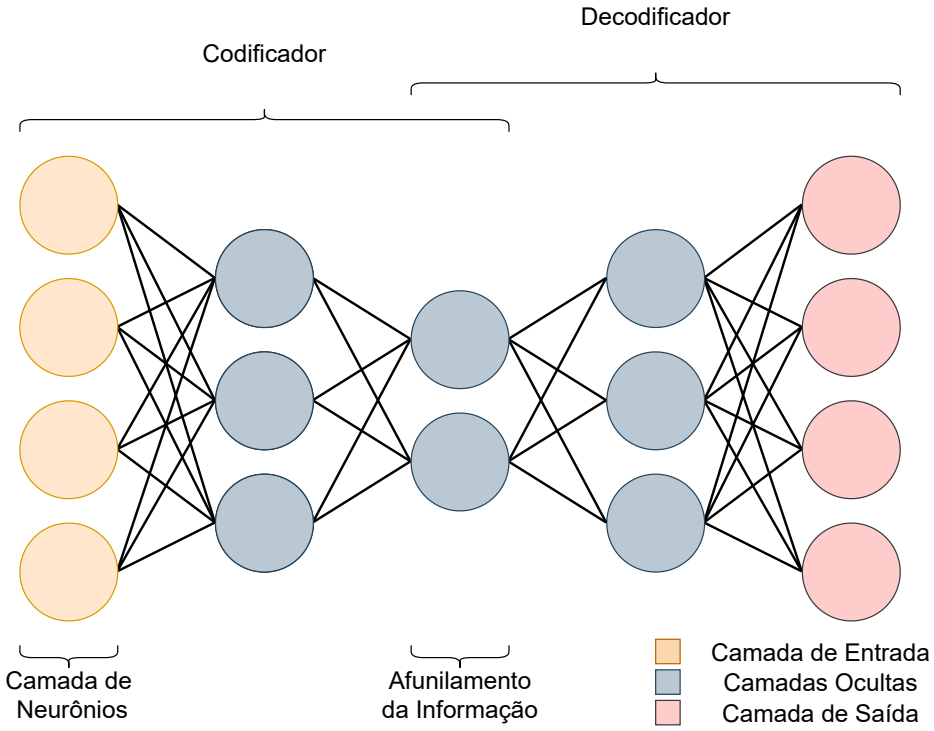

Figura 5 – Rede Neural *autoencoder*. Fonte: O próprio autor.

3 REDES NEURAIS CONVOLUCIONAIS

3.1 Introdução

Redes convolucionais têm como principal característica o processo de convolução de valores em algumas camadas, que faz a extração de características importantes ao mesmo tempo que simplifica os dados, tornando-os menos custosos de se trabalhar [20]. Essa simplificação viabiliza o uso de redes neurais para processar imagens, já que, dependendo da resolução da entrada, seria necessário uma grande quantidade de neurônios além das conexões entre eles para uma rede neural comum ser utilizada [21][20]. Em uma rede profunda, várias camadas de convolução são utilizadas em sequência. Nas primeiras camadas, elementos simples como cantos e bordas são detectados, destacados e enviados para a próxima camada que, por sua vez, extrairá informações mais importantes como círculos e quadrados. Com camadas o suficiente, torna-se possível detectar padrões cada vez mais complexos como olhos, boca e nariz, seguindo para o rosto completo de um ser humano [20][59][60]. É claro que, apesar do foco em processamento de imagens, esse tipo de rede neural também pode ser utilizado em outras aplicações, incluindo detecção de anomalias, como mencionado na seção Aplicações e Estudos.

Apesar do foco em processamento de imagens, esse tipo de rede neural também pode ser utilizado para detectar anomalias como mencionado nas seções seguintes

3.2 A convolução

O processo de convolução de duas dimensões utiliza uma pequena matriz chamada de *kernel* como filtro para extrair características de uma outra matriz de entrada. Esse filtro é um dos parâmetros treináveis da rede, e pode ser comparado ao neurônio da rede neural artificial tradicional, tal que uma camada convolucional pode conter mais de um *kernel*. Cada elemento da matriz filtro pode ser alterado individualmente e são comparáveis aos pesos das redes tradicionais. O *kernel* também possui um valor de viés, que também pode ser modificado e será somado ao valor final da convolução.

O algoritmo passa por cada célula da matriz inicial fazendo o produto escalar daquela região com o filtro e armazena o resultado em uma nova matriz. Essa nova matriz é menor ou igual à inicial e tem valores modificados de acordo com o filtro. Esse processo está representado na Figura 6. Em seguida, é aplicada uma função de ativação não linear em cada um dos elementos da nova matriz [21]. Em uma camada com mais de um filtro, a matriz original recebe cada um dos filtros e uma nova matriz é gerada para cada um. Esse conjunto de matrizes resultantes é chamado de *feature map*.

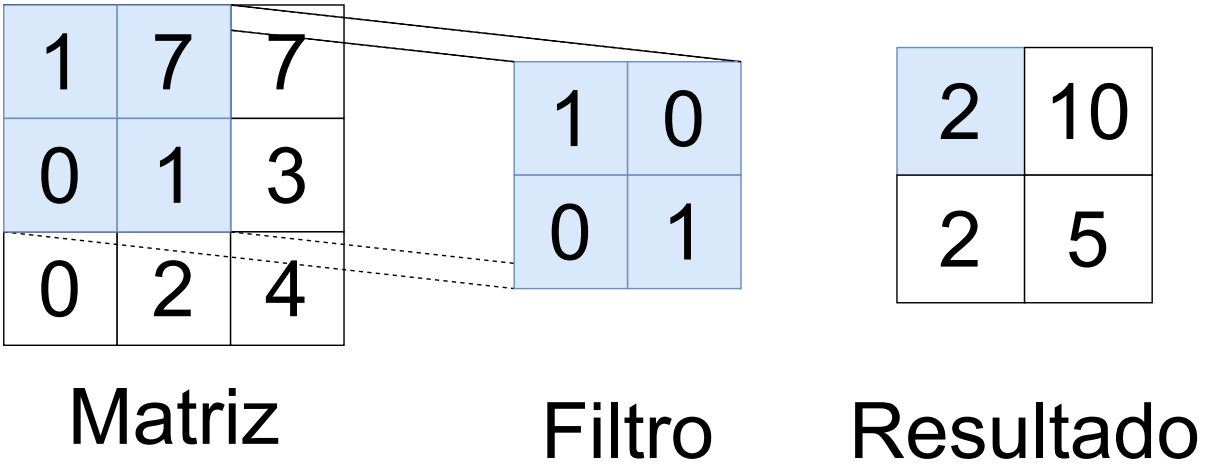

Figura 6 – Exemplo de convolução. Fonte: O próprio autor.

Em uma rede convolucional, é comum a existência de mais de uma camada de convolução. Assim, as camadas seguintes à primeira receberão como entrada a saída da camada anterior. Nesse caso, a camada posterior receberá uma matriz para cada *kernel* da camada anterior. Por isso, cada *kernel* da nova camada será composto por K matrizes, em que K é o número de *kernels* da camada anterior. O resultado de cada filtro é somado para gerar a matriz resultante.

3.3 Passo (*Stride*)

Além do tamanho dos *kernels*, um outro hiperparâmetro que pode ser alterado é o passo (*Stride*) da convolução. O passo é o número de células que o filtro "anda" para a convolução. Normalmente o valor 1 é utilizado, evitando ao máximo, mas não completamente, a diminuição de dimensão da matriz resultante em relação à original. Esse valor também maximiza a sobreposição de informações, o que significa que uma mesma célula influenciará o máximo de células resultantes possível [21]. A equação (3.1) exemplifica o cálculo das dimensões resultantes (O) sem que haja preenchimento (explicado na próxima seção) em que N é o tamanho da matriz original, F é o tamanho do filtro e S é o passo [20].

$$O = 1 + \frac{N - F}{S} \tag{3.1}$$

Como é possível notar pela equação (3.1), para valores de passo maiores do que 1, a dimensão resultante diminui ainda mais do que o valor já reduzido pelo filtro. Além disso, a sobreposição de informações utilizadas diminui, chegando a deixar de existir quando o passo se torna maior ou igual ao tamanho do filtro.

3.4 Preenchimento (*Padding*)

Ao executar a convolução com filtros de tamanho maior que 1, é preciso que o filtro "encaixe" na matriz de forma que nenhuma de suas células fique sem um valor para ser multiplicado. Por isso, a posição central do *kernel* quando na matriz original não pode ficar em sua borda. Isso implica na perda de informações dos elementos das extremidades da matriz, e na diminuição da dimensão dos dados resultantes. Esse efeito pode ser evitado utilizando o preenchimento de bordas (*Padding*). Nesse processo, que ocorre antes da convolução, a matriz de entrada tem suas dimensões aumentadas com células preenchidas por 0s, possibilitando que o filtro inicie a convolução na célula que antes era a extremidade da matriz original. É possível modificar a equação (3.1) para receber o parâmetro P, indicando o número de camadas adicionadas na borda para a seguinte equação: (3.2)[20].

$$O = 1 + \frac{N + 2P - F}{S} \tag{3.2}$$

3.5 *Pooling*

Depois da convolução, pode ser feito o processo de *pooling*, que consiste em diminuir o tamanho das matrizes resultantes. Para isso, costuma-se dividir os dados resultantes em regiões e armazenar apenas o maior valor de cada uma, chamado de *Max Pooling*; ou somente a média, chamado de *Average Pooling* em uma nova matriz de tamanho reduzido. Assim, o processo de mais convoluções pode começar a se repetir até que não hajam mais camadas de convolução na rede neural.

O principal objetivo da operação de *pooling* é diminuir a complexidade dos dados [20]. Mesmo que exista uma perda de informação, a diminuição da complexidade ajudará na performance da rede já que cada *kernel* da operação de convolução aumenta a profundidade dos dados. A camada de *pooling* diminui as dimensões de altura e largura sem alterar a profundidade dos dados, isto é, a operação não altera a dimensão criada pelos *kernels*, apenas as dimensões originais da entrada [21]. A operação é utilizada quando apenas a presença da informação importa, e não sua posição; diante disso, *max pooling* é o mais frequentemente utilizado, já que destaca, sem alterar, a informação mais acentuada [20].

Normalmente, o tamanho da região vai de 2x2 até 3x3 em operações de duas dimensões já que valores maiores do que esses poderiam diminuir consideravelmente o desempenho da rede por resultar em uma perda significativa de dados [21]. Por exemplo, ao trabalhar com uma matriz 16x16, tamanho do *pooling* 2x2 e passo 2, o tamanho de cada dimensão será dividido por 2, resultando em uma matriz 4x4; uma redução de 75% da original [21].

3.6 Camada *Flatten*

A camada *flatten* tem como objetivo formatar os dados da matriz resultante para um único vetor para possibilitar sua utilização em camadas seguintes que não são convolucionais. Dessa forma, após a extração de características das camadas convolucionais, outras camadas conseguem utilizar os dados obtidos seja para decidir sobre algo ou para continuar a extrair características de outra forma.

3.7 Camada Densa

Por fim, após camadas de convolução e *pooling*, uma rede convolucional pode utilizar camadas densas de neurônios completamente conectados. A complexidade dessas camadas é consideravelmente menor do que precisaria ser caso não houvesse a diminuição das dimensões da matriz de entrada. Essas camadas são responsáveis por interpretar as características extraídas pelas convoluções e responder problemas como classificações da mesma forma que são utilizados em redes neurais artificiais tradicionais [61].

3.8 Aplicações em estudos

3.8.1 Reconhecimento facial

Reconhecimento facial é extremamente relevante devido ao grande número de possíveis aplicações tanto comerciais quanto de segurança. Alguns exemplos incluem detecção de rostos em imagens ao tirar fotos, reconhecimento de identidade para desbloqueio de aparelhos e até identificação de indivíduos em câmeras de vigilância. O método se destaca dentre outros tipos de biometria por não necessariamente necessitar da colaboração do humano [62] Por isso, o assunto já é amplamente estudado e já foram desenvolvidas técnicas sobre o problema [63]. Dentre elas, redes neurais convolucionais se provam eficazes, como demonstram os estudos abaixo.

Em [64], inicialmente apresentou como problema a falta de bancos de dados públicos que contenham mais de um milhão de imagens. Durante o desenvolvimento do trabalho, um novo banco que utiliza imagens de atores foi construído. Para isso, foram utilizados a lista de celebridades do *Internet Movie Data Base* (IMDB) e a pesquisa do Google e Bing para coletar amostras. Para remover possíveis elementos incorretos, uma rede convolucional foi treinada reconhecer imagens de cada pessoa, separando os dados em blocos que foram, em seguida, avaliados por humanos. Finalizada a preparação dos dados, foi desenhada a rede com 16 camadas convolucionais, cada uma seguida de uma camada de *Rectified Linear Unit* ($ReLU$) com exceção da última, que utiliza *softmax*. Para o treinamento, algumas técnicas foram utilizadas; dentre elas, decaimento da taxa de aprendizado, inversão horizontal e apenas partes das imagens como entrada. Como

resultado, o modelo apresenta taxas de precisão acima de 97% em bases de dados como *Labeled Faces in the Wild* (*LFW*) e *Youtube Faces*.

Em [62], também foi proposto um reconhecimento de identidade. A base de dados utilizada foi a *Georgia Tech*, que possui imagens de 50 indivíduos capturadas em duas ou três sessões em diferentes horários com 15 imagens no total para cada pessoa. O estudo avalou diferentes técnicas de pré-processamento de cores, uma delas separa a imagem em 3 cores, a outra converte a imagem para uma única cor, chegando à conclusão de que 3 imagens sempre superam uma só. Também foram testadas diferentes resoluções para as imagens de entrada; intuitivamente, a maior resolução teve melhor desempenho. Para a rede convolucional, foram utilizadas 5 camadas convolucionais com 2 camadas de normalização de valores, uma após a primeira convolução e outra após a última. Camadas de redução de dimensão (*Pooling*) foram utilizadas após a primeira, segunda e terceira camadas convolucionais. Terminada a extração de características das convoluções, os dados foram enviados à uma rede neural softmax para classificar os elementos. A acurácia foi de 94.8% para o pré-processamento de 3 imagens com resolução de 64x64, um resultado satisfatório de acordo com o estudo.

3.8.2 Reconhecimento de fala e Processamento de Linguagem Natural

Ter uma máquina que entende e responde à fala humana é um sonho da ficção que já se tornou realidade com os assistentes pessoais. Isso só é possível devido aos avanços nas áreas de reconhecimento de fala, que traz a capacidade de transformar voz em texto, e o processamento de linguagem natural, que possibilita a classificação e interpretação para responder. Além disso, interpretação de texto e extração de informações em aplicações possibilitam que mecanismos de pesquisa como Google analisem milhares de páginas para encontrar respostas pertinentes [65]. Devido à relevância do contexto e sequencialidade para aplicações do gênero, redes neurais recorrentes se destacam sobre as convolucionais [66][67].

Em [68], foi proposto um modelo que utiliza múltiplas camadas convolucionais em sequência para processar texto. O foco é utilizar a profundidade da rede para processar texto à partir da unidade mais simples: o caractere. Para isso, foram feitos testes com até 49 camadas convolucionais. Também foram testados diversos bancos de dados com tarefas diferentes. Classificação de notícias em inglês e chinês, classificação de tópicos do yahoo respostas e classificação de críticas. O modelo teve melhor desempenho em conjuntos de dados maiores mesmo para profundidades menores. Finalmente, o melhor modelo era composto por 29 camadas já que qualquer número acima desse aumentava a taxa de erro. Por fim, o estudo concluiu que o modelo consegue superar o estado da arte em quase todos os bancos de dados testados e que o processamento de pequenos textos tem propriedades parecidas com o processamento de imagens. Os autores acreditam que a performance de

processamento de texto pode ser melhorada com modelos mais profundos.

3.8.3 Detecção de Anomalias

Problemas de detecção de anomalias podem ser abordados de diversas maneiras como explicado no capítulo anterior. Normalmente, ao utilizar aprendizado profundo para resolver esse tipo de problema, redes neurais recorrentes são preferidas às redes convolucionais por possuírem desempenho superior [69]. Em [69], foram testados 3 modelos com profundidades diferentes. O mais raso, com apenas 1 camada convolucional e 1 de *max pooling*, empatou com os dois outros modelos com 2 e 3 camadas convolucionais em 2 conjuntos de dados testados, enquanto venceu em um terceiro conjunto. Ainda assim, a melhor rede convolucional se mostrou pior em comparação com modelos estudados pelos autores anteriormente.

Contudo, em [22] um modelo foi proposto para detectar ataques distribuídos de negação de serviço em redes definidas por *software*. O conjunto de dados utilizado para testes foi o CICIDS2017, que é composto por 5 dias diferentes de tráfego e 8 dias no total, sendo um deles completamente livre de ataques. Dentre os outros dias, ataques como *DDoS*, *DoS*, *PortScan* estão presentes [70]. O modelo discrimina o tráfego como anômalo ou normal. Os autores utilizam a biblioteca Keras para combinar dois classificadores de arquiteturas iguais em um só. Isso possibilita que o modelo alcance 99.51% de precisão e 99.74% de revocação, resultado que justifica o uso de redes neurais convolucionais e conclui que a combinação de classificadores é um facilitador em detecção de *DDoS* tanto pelos bons resultados quanto pela diminuição no tempo de testes.

4 TRABALHOS RELACIONADOS

4.1 Trabalhos em ambientes de redes tradicionais

Assis et al. [15] apresentaram um sistema de detecção de anomalias que utiliza sete dimensões de fluxo para classificar o tráfego como anômalo ou normal. Nesse trabalho foi criada uma assinatura que descreve o comportamento normal das dimensões de fluxo e identifica possíveis anomalias. A técnica utilizada é uma modificação do método estatístico *Holt Winters*, utilizado para previsão de séries temporais.

Proença et al. [71] apresentaram um experimento que coletou dados da Universidade Estadual de Londrina para gerar uma assinatura digital da rede. Dois tipos de assinaturas foram criadas; a primeira separa assinaturas para cada dia da semana, enquanto a segunda calcula uma assinatura média para os dias da semana, outra para sábados e mais uma para domingos. Também foram definidos três limiares diferentes que apresentam diferentes níveis de sensibilidade com o intuito de reduzir falsos positivos. Por fim, regras foram estabelecidas para que o sistema decida se deve ou não soar o alarme. Dessa forma houve uma redução significativa nos falsos positivos.

Pena et al. [31] utilizaram lógica paraconsistente para decidir se há anomalia ou não sobre o tráfego na rede baseado na assinatura digital gerada a partir de dados previamente estudados. A lógica paraconsistente tem a função de compensar a imperfeição que pode existir nos dados, já que sua coleta vem de uma rede real sem a utilização de filtros. O estudo também apresenta diferentes formas de gerar a assinatura digital, como séries temporais e heurística de colônia de formigas.

No trabalho de Kwon et al. [69], foram testados 3 modelos de redes convolucionais para detectar anomalias em rede. A principal diferença entre os modelos era sua profundidade. Para os testes, foram utilizados três conjuntos de dados: *NSL-KDD*, *Kyoto Honeypot* e *MAWILab*. Ao fazer testes, a rede neural menos profunda foi a que demonstrou melhor performance no conjunto *NSL-KDD*, enquanto houve um empate nos testes dos conjuntos *MAWILab* e *Kyoto Honeypot*. Em seguida, a performance da rede rasa foi comparada com outros tipos de rede neural mas não apresentou resultados melhores que redes *LSTM*, por exemplo.

No trabalho de Dromard et al. [72], foi proposto um sistema de detecção de anomalias que classifica o tráfego em um dispositivo utilizando um algoritmo de *clustering*. Para isso, é feita uma análise que leva em consideração os últimos 15 segundos de tráfego e até 17 características do fluxo. Dessa forma, o sistema conseguiu detectar anomalias em um intervalo de menos de meio segundo de sua ocorrência.

No trabalho de Aygun et al. [73], foi apresentada uma abordagem de detecção de anomalias utilizando redes neurais *autoencoder*. Para fazer a detecção, a rede neural é primeiro treinada para reconstruir o tráfego normal de forma que, quando receber tráfego anômalo, seu erro de reconstrução será mais alto que o normal. Assim, basta escolher limiares de decisão para decidir sobre a existência de anomalias

No trabalho de Naseer et al. [74], foram testadas redes neurais do tipo convolucionais, *autoencoders* e recorrentes para detecção de anomalias. O conjunto de dados de teste foi o NSL-KDD e os melhores modelos foram os *LSTM* e CNN enquanto o *autoencoder* não se mostrou tão eficiente.

Em Bereziski et al. [75], foi apresentado um modelo que utiliza dados de entropia para apontar anomalias. Nesse modelo, os limiares superiores e inferiores são calculados a partir de uma janela de tempo do tráfego sem anomalias. Assim, ao ser apresentado ao tráfego anômalo, os limiares serão transgredidos.

4.2 Tabalhos em ambientes de redes definidas por software

Scaranti et al. [25] apresentaram um sistema de detecção de intrusão com funcionamento inspirado no sistema de defesa do corpo humano que percebe e age contra organismos desconhecidos. Analogamente, o sistema proposto percebe o tráfego anômalo à partir do treinamento utilizando apenas tráfego comum da rede. Além disso, seu módulo de mitigação reconhece as anomalias conhecidas e identifica agentes maliciosos para bloquear ataques.

Já no trabalho de Assis et al. [19], foi destacada a importância da rápida atuação dos sistemas de detecção de intrusão e mitigação devido ao elevado crescimento do volume de informações enviadas diariamente pela rede. No estudo foram testadas 3 abordagens de detecção, sendo elas redes neurais, transformada de *wavelet* e otimização por enxame de partículas.

Novaes et al. [38] propuseram um trabalho de detecção e mitigação de ataques de negação de serviço e *portscan* em ambientes de redes definidas por *software*. O sistema prevê o comportamento normal do tráfego e define margens de decisão; aplica lógica difusa para determinar se há ou não anomalias utilizando como parâmetro o tráfego predito e aplica políticas de mitigação.

No trabalho de Hamamoto et al. [1], foi proposto um sistema que monitora o tráfego da rede enquanto gera comportamento esperado, detectando anomalias. O modelo apresentado utiliza algoritmo genético para caracterizar o tráfego e estabelecer limiares e lógica difusa para decidir sobre a presença de anomalia utilizando os dados calculados anteriormente. O método possui altas taxas de precisão enquanto mantém baixas taxas de falsos positivos.

Assis et al. [33] apontaram como um dos problemas de dispositivos *IoT* a falta de segurança e sua utilização em ataques *DDoS*. Para combater o problema, o estudo propôs um sistema de segurança que analisa dimensões de fluxo e utiliza redes neurais convolucionais para detectar anomalias.

No trabalho de Carvalho et al. [76], foi proposto um ecossistema *SDN* projetado para detectar e mitigar ameaças. Para detecção de anomalias, utiliza os dados de fluxo coletados e uma heurística de colônia de formigas utilizada em [36]. Para mitigação, políticas foram determinadas para tomar ações dependendo dos dados coletados pela *SDN*.

Haider et al. [22] apresentaram uma solução para ataques *DDoS* em redes definidas por *software* utilizando redes neurais convolucionais. Para testes foi utilizado o dataset *CICIDS2017*. O modelo alcançou taxas de precisão de 99.48% ao detectar ataques enquanto mantém baixa complexidade computacional.

Qin et al. [77] apresentaram um modelo que utiliza *CNN* e *RNN* para detectar anomalias em redes definidas por *software*. Nesse trabalho, foi utilizado o programa *mininet* para emular uma *SDN* e *Python* para produzir pacotes. Foram gerados ataques de *portscan*, *syn flood* e "chute" de senha de *FTP*. Além disso, também foram feitos testes com o conjunto de dados CTU-13

O trabalho de Fernandes et al. [2] é uma pesquisa sobre detecção de anomalias que apresenta detalhadamente conceitos utilizados no ramo, dentre eles: formas de categorização de anomalias; tipo de dados coletados para sua detecção; tipos de sistemas de detecção e técnicas e métodos para detecção.

Osanaiye et al. [32] apresentaram conceitos e soluções de defesa contra ataques *DDoS* à servidores. Para detectar ataques, foram propostos métodos como *change point detection* analisando o tráfego e comparando com padrões obtidos anteriormente. Contudo, uma nova dimensão de fluxo é utilizada, o *packet interarrival time*.

Dentre os estudos analisados, é possível confirmar que redes definidas por software são um paradigma que soluciona diversos problemas e se tornará cada vez mais dominante. A Tabela 1 e a Tabela 2 a seguir apresentadas contêm um resumo dos trabalhos mencionados neste capítulo.

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

Autor	**Dados Utilizados**	**Ambiente**	**Principais Características**
Assis, M. V. O. et al. [15]	Dados reais da UEL	Redes Tradicionais	Sistema de detecção de anomalia utilizando o método holt winters e 7 dimensões de fluxo para criar assinatura digital do tráfego.
Proenca, M. L. et al. [71]	Dados Reais da UEL	Redes Tradicionais	Gera duas categorias de assinatura da rede e três limiares de decisão para detecção de anomalias na rede além de regras para decisões.
PENA, E. H. et al. [31]	Dados Reais da UEL	Redes Tradicionais	Assinatura digital e lógica paraconsistente para detectar anomalias. Séries temporais e heurística de colônia de formigas para gerar assinatura.
Kwon, D. et al. [69]	NSL-KDD, Kyoto Honeypot, MAWILab	Redes Tradicionais	Testes com 3 modelos de redes convolucionais em 3 conjuntos de dados e comparação com outros modelos de redes neurais.
Dromard, J. et al. [72]	Dados Reais Provedor de *Internet* e MAWILab	Redes Tradicionais	Algoritmo de clusterização, que detecta anomalias em tempo real.
Aygun, R. C. et al. [73]	NSL-KDD	Redes Redes Tradicionais	Solução com *autoencoders* e limiares de decisão
Naseer, S. et al. [74]	NSL-KDD	Redes Tradicionais	Redes neurais convolucionais, redes *LSTM* e *autoencoders*
BEREZISKI, P. et al. [75]	Dados Emulados	Redes tradicionais	Utiliza características de entropia, e uma janela deslizante para estabelecer limiares de decisão
Scaranti, G. F. et al. [25]	Emulados e *CiCDDoS2019*	*SDN*	Sistema de detecção de intrusão utiliza treinado apenas com tráfego comum.
Assis, M. V. O. et al. [19]	Dados Emulados	*SDN*	Testes com 3 abordagens de detecção: redes neurais, transformada de *wavelet* e otimização por enxame de partículas.
Novaes, M. P. et al. [38]	Emulados e *CICDDoS* 2019	*SDN*	Detecção e mitigação de ataques de negação de serviço e *portscan* em *SDN*. Usa lógica difusa para determinar se há ou não anomalias.
HAMAMOTO, A. H. et al. [1]	Dados Reais da UEL	*SDN*	Utiliza algoritmo genético para caracterizar o tráfego e estabelecer limiares, ao mesmo tempo que utiliza lógica difusa para decidir sobre anomalias.

Tabela 2 – Resumo trabalhos relacionados (Continuação)

Autor	Dados Utilizados	Ambiente	Principais Características
Assis, M. V. et al. [33]	Emulados e *CicDDoS* 2019	*SDN*	Propõe um sistema de segurança que analisa dimensões de fluxo e utiliza redes neurais convolucionais para detectar anomalias.
CARVALHO, L. F. et al. [76]	Dados Emulados	*SDN*	Propõe um ecossistema *SDN* para detectar e mitigar ameaças. Para detecção de anomalias, utiliza uma heurística de colônia de formigas.
Haider, S. et al. [22]	*CICIDS2017*	*SDN*	Utiliza CNN para detectar ataques *DDoS* em *SDN*.
Qin, Y. et al. [77]	Dados Emulados, CTU-13 e CSIC	*SDN*	Utiliza redes convolucionais e recorrentes
FERNANDES, G. et al. [2]	-	Redes Tradicionais e *SDN*	Uma pesquisa sobre detecção de anomalias. Apresenta formas de categorização de anomalias; tipo de dados coletados para sua detecção; tipos de sistemas de detecção e técnicas e métodos para detecção.
OSANAIYE, O. et al. [32]	-	*SDN*	Apresenta conceitos e soluções de defesa contra ataques *DDoS* à servidores.

5 ESTUDO DE CASO

Neste capítulo, um estudo prático é demonstrado sobre um sistema que recebe o tráfego na rede a cada segundo e decide sobre a existência de uma anomalia naquele instante. Serão descritas informações sobre os dados, a arquitetura da rede, seu treinamento, método de detecção e resultados. O sistema está ilustrado na Figura 7. A configuração utilizada para o desenvolvimento do trabalho utilizou sistema operacional Windows 10, 32GB de memória ram, um processador AMD Ryzen 1700X e placa de vídeo Nvidia RTX 2060.

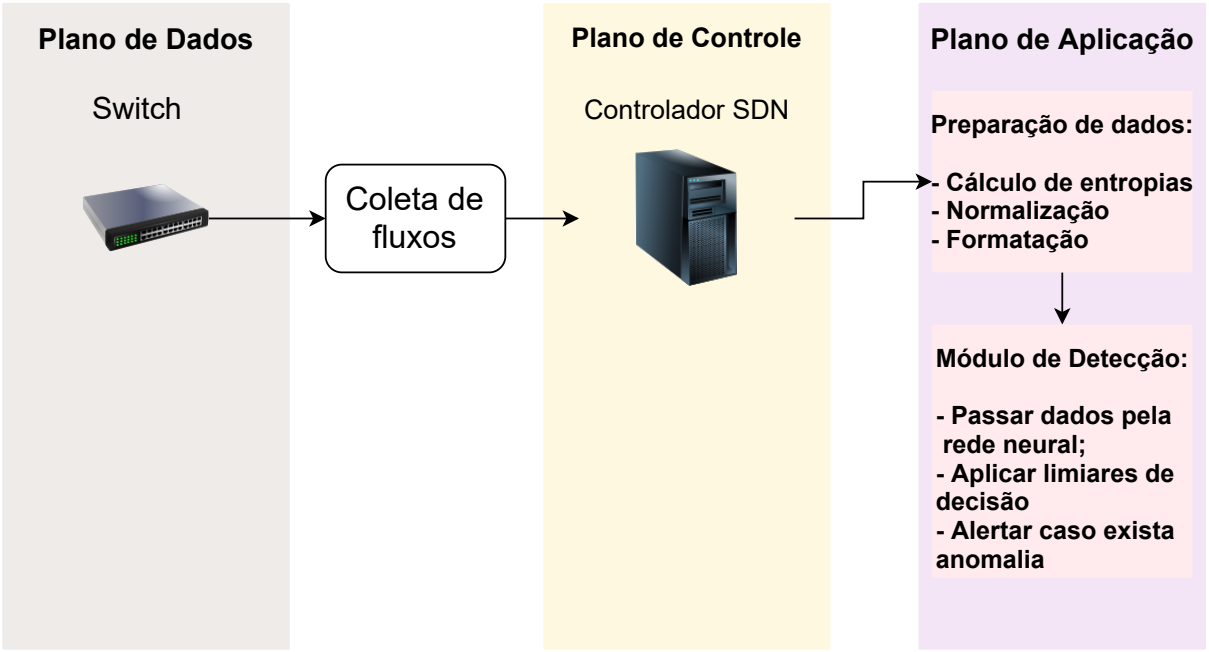

Figura 7 – Sistema de Detecção de Anomalias. Fonte: o próprio autor

5.1 Dados Utilizados

5.1.1 Pré-processamento: a Entropia de Shannon

Entropia é um conceito normalmente utilizado na química e física que representa o grau de desordem ou aleatoriedade de um sistema [78]. Entretanto, o termo também é utilizado no ramo da teoria da informação para quantificar a incerteza de alguma variável. Por exemplo, a incerteza do resultado de um dado de seis lados é menor do que a de escolher uma carta em um baralho. O cálculo da entropia utiliza a probabilidade de que cada evento aconteça, levando em consideração tudo que é necessário para sua realização,

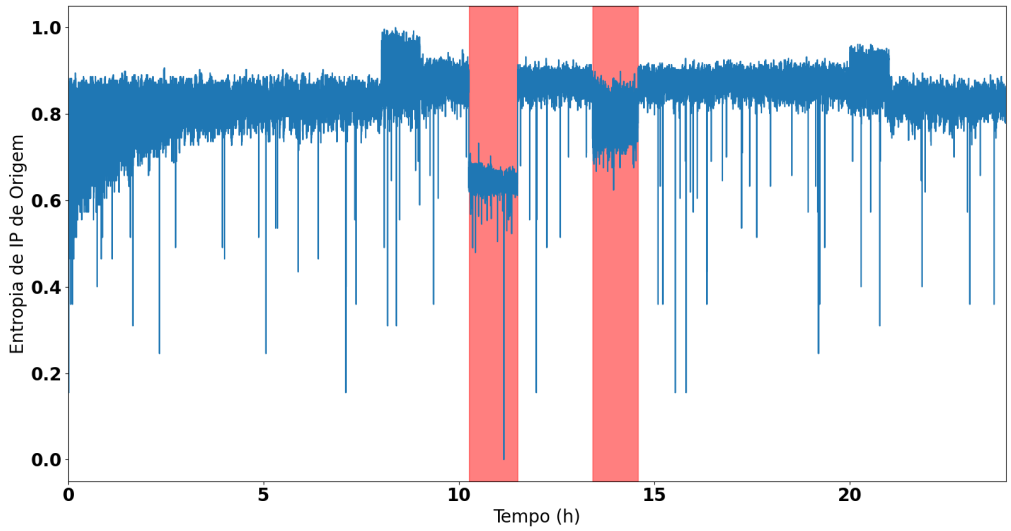

Figura 8 – Entropia de *IP* de origem medida durante um dia. Fonte: O próprio autor.

utilizando a fórmula descrita na Equação 5.1 [79]:

$$H = -\sum_{i=1}^{N} p_i \log_2(p_i) \tag{5.1}$$

Em que i representa cada evento individual, pi a probabilidade desse evento acontecer, e N a quantidade de eventos possíveis. Nessa fórmula, H resultará em um número baixo, caso a probabilidade de um único evento seja extremamente alta comparado aos demais, enquanto será mais alta caso existam vários eventos com probabilidades similares.

Em redes de computadores é comum a ocorrência de diversos usuários diferentes fazendo requisições a um servidor ao mesmo tempo. Nesse caso, pode-se dizer que a entropia de endereços é elevada, já que, a qualquer momento, algum usuário diferente pode fazer uma requisição. Sabendo disso, no caso de uma ocorrência de ataque de negação de serviço, um mesmo conjunto de endereços estará enviando um maior volume de requisições que o normal. Como resultado, a entropia do sistema deve diminuir, já que é mais provável que uma requisição venha dos endereços maliciosos do que de um usuário comum [38]. É possível observar na Figura 8 a diminuição da entropia de endereços *IP* de origem nos momentos destacados em vermelho, quando ocorrem ataques de negação de serviço e *portscan* respectivamente.

5.1.2 O conjunto de dados

Os dados foram disponibilizados pelo Grupo de Redes Orion, da Universidade Estadual de Londrina e também foram utilizados no estudo [25] que os descreve. Eles

foram gerados utilizando um emulador de redes definidas por *software* chamado *mininet*[1] para criar uma estrutura de 6 *switches* em uma topologia de árvore. Nessa topologia, o primeiro *switch* é a raiz da 'árvore' e conecta os demais 5 *switches* que estão conectados cada um a 12 *hosts*. O tráfego dessa rede é emulado utilizando a biblioteca Scapy[2], que cria e envia pacotes pela rede. Conforme os horários do dia, um volume de pacotes diferente é gerado como ocorre em uma rede real. Por fim, o *software hping3*[3] é utilizado para simular os ataques de *DDoS*, programando diversos *hosts* para enviar múltiplas requisições a um único *host* de destino. Ataques de *porstscan* também são simulados entre dois *hosts* utilizando o próprio Scapy.

No estudo [25], são registrados 3 dias de dados. O primeiro representa o tráfego normal da rede na ausência de anomalias. Cada um dos outros dias contém exatamente dois ataques, sendo um de *DDoS* e outro de *portscan* em momentos diferentes. Para este trabalho, mais um dia de dado emulado foi disponibilizado, totalizando quatro dias com dois ataques cada em horários e durações diferentes.

Os dados registram as seguintes dimensões de fluxo: *bit* por segundo; pacotes por segundo; entropia de *IP* de origem; entropia de porta de origem; entropia de *IP* de destino e entropia de porta de destino. Cada dia de dados é dividido em 86400 amostras, cada uma representando 1 segundo de fluxo da rede para cada dimensão de fluxo. Assim, uma matriz 86400 x 6 foi gerada para cada dia, de forma que cada linha representará 1 segundo do dia com cada coluna representando uma das dimensões. No gráfico da Figura 9 está ilustrado o primeiro dia de dados após ser normalizado utilizando a biblioteca *Scikit-learn*[4]. Já na Figura 10 e Figura 11, estão representados os dias com ataques destacados em vermelho. Na Figura 10, o primeiro ataque é o de *DDoS* e o segundo de *portscan*. Já na Figura 11, o primeiro ataque é o de *portscan* e o segundo de *DDoS*. A normalização utilizada para as figuras é chamada de *Standard Scaler*, contudo, no treinamento e teste, a normalização utilizada foi a *Min Max Scaler* com o intervalo de 0 a 1. A normalização é feita para aproximar os valores das dimensões de forma que cada um tenha a mesma influência no treinamento da rede neural.

[1] http://mininet.org/
[2] https://scapy.net/
[3] http://www.hping.org/
[4] https://scikit-learn.org/

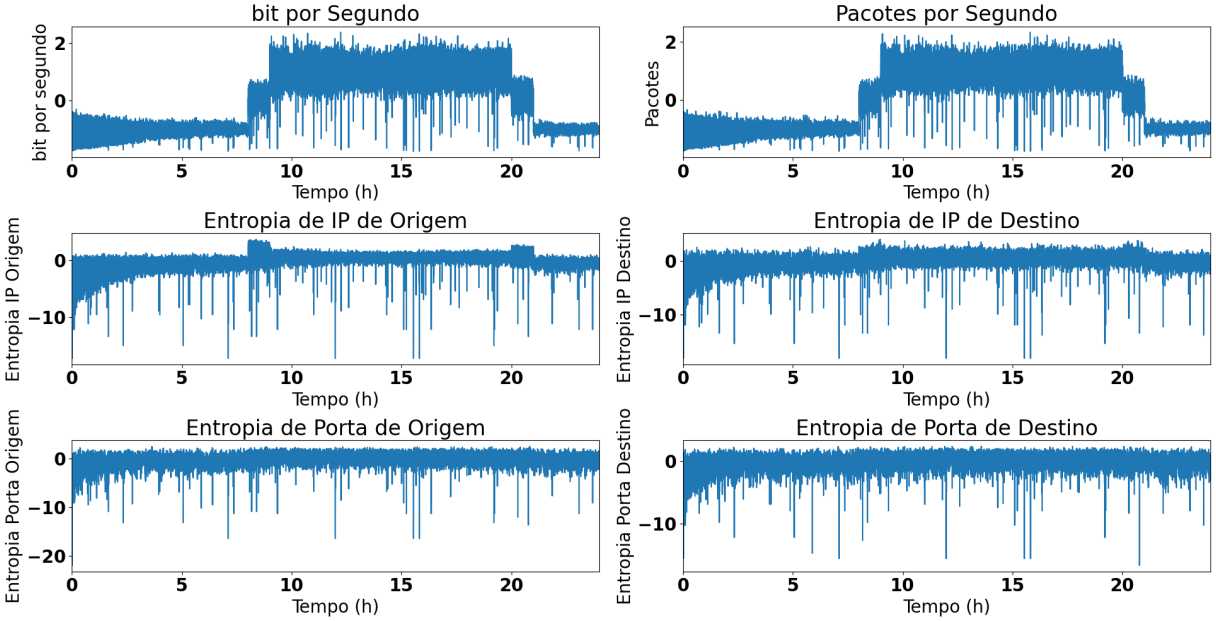

Figura 9 – Primeiro dia de dados emulados, sem ataque. Fonte: o próprio autor

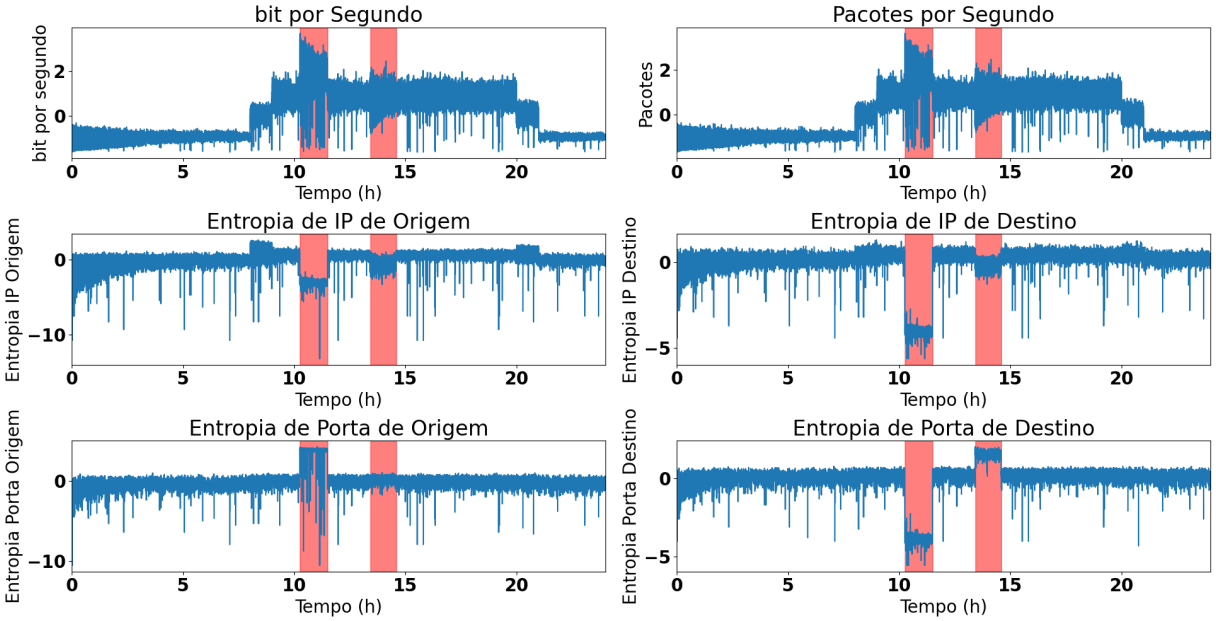

Figura 10 – Segundo dia de dados emulados, com ataque. Fonte: o próprio autor

5.2 A Rede Neural

A rede neural foi criada utilizando a biblioteca Keras[5] para *Tensorflow* para desenvolvimento de aplicações de *deep learning* da linguagem *Python*. O modelo possui uma camada de entrada que recebe um vetor de 6 posições, fazendo a análise a cada segundo. Em seguida, uma camada de convolução de uma dimensão com 8 filtros de tamanho 1x2, resultando em uma saída de 8 vetores de dimensões 1 x 5. Na sequência, uma camada

[5] https://keras.io/

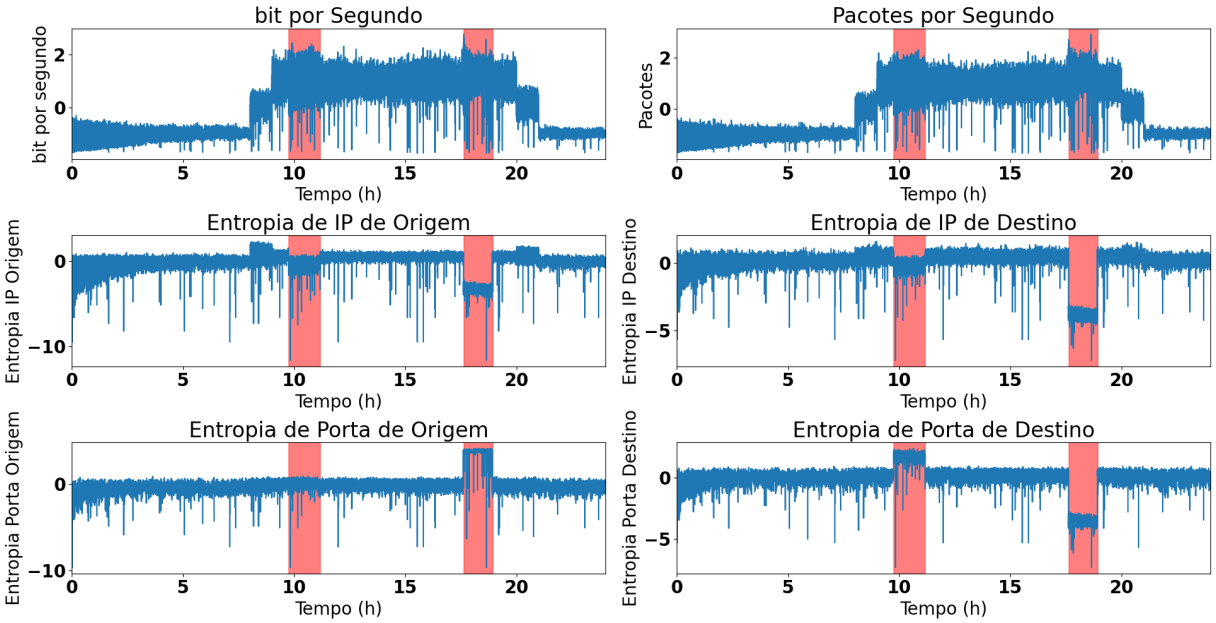

Figura 11 – Terceiro dia de dados emulados, com ataque. Fonte: o próprio autor

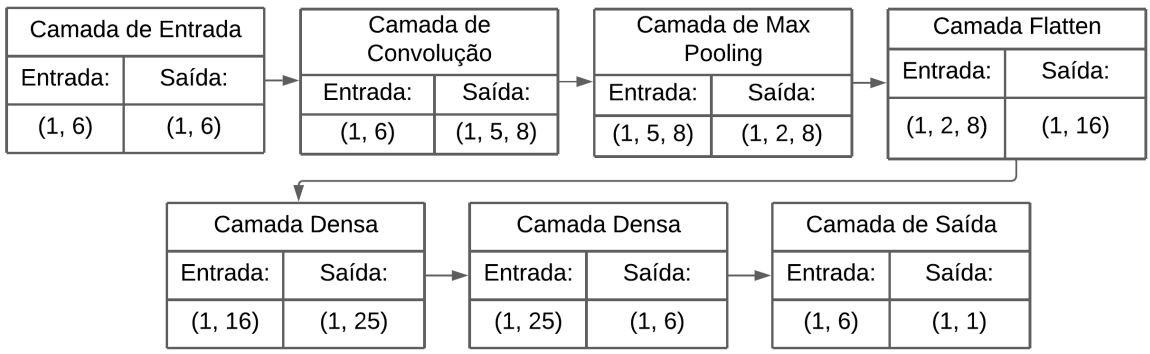

Figura 12 – Fluxo de dados da rede neural. Fonte: O próprio autor.

de *Max Pooling* com filtro de tamanho 2, passo 2 e nenhum preenchimento reduzem o tamanho dos dados para 8 vetores de dimensões 1 x 2. A camada *flatten* recebe os vetores e os agrupa em um único vetor de 16 posições para ser enviado para as camadas densas. São 3 camadas em sequência, uma com 25 neurônios, outra com 6 e a última com 1. Todas as camadas, utilizam como função de ativação a *Rectified Linear Units* (*ReLU*), com exceção da última que faz uso da *sigmoid* para que a rede tenha uma saída dentro do intervalo de 0 a 1. A função *relu* foi escolhida por ser mais eficiente e por seu gradiente possibilitar um aprendizado mais rápido [47][46]. Na Figura 12 é possível ver o fluxo dos dados.

A imagem da função *sigmoid* vai de 0 a 1 e nesse modelo representa o valor previsto do próximo segundo de tráfego na rede. Para este modelo, a dimensão de fluxo escolhida foi a entropia de *IP* de destino, que foi normalizada para corresponder ao intervalo da

função sigmoid.

5.3 Treinamento, validação e limiares de decisão

Para realizar o treinamento da rede neural, foi utilizado o dia sem ataques. A entrada da rede recebe as 6 dimensões de fluxo e o rótulo de treinamento foi a dimensão de fluxo entropia de *IP* de destino do segundo seguinte. Por isso, o último segundo da entrada e o primeiro segundo da saída foram removidos do conjunto de dados, totalizando 86399 entradas. Dessa forma, a saída da rede será uma previsão de 0 a 1 da entropia de *IP* de destino normalizada. Para validação, foi utilizado o intervalo sem ataques do primeiro dia de ataque. O treinamento foi feito por 30 épocas.

O cálculo para decisão sobre a existência de anomalia foi realizado da seguinte forma: primeiro o tráfego previsto pela rede neural com a entrada de um dia sem anomalia é calculado; em seguida, é feita a previsão utilizando os dados com ataque; por fim, o erro absoluto da diferença entre os dois resultados é calculado, e é considerado anômalo o tráfego cujo erro excede o limiar de decisão. O cálculo do erro absoluto está representado na Equação 5.2, em que v_{sa} é o valor previsto no dia sem anomalias e v_p é o valor previsto dos dados com ataque.

$$erro = |v_{sa} - V_p| \tag{5.2}$$

Para escolher os limiares de decisão, foram estudadas duas possibilidades. A primeira visou maximizar a medida F1 representada na Equação 5.5. Essa medida utiliza as métricas de precisão, representada na Equação 5.3, e revocação, apresentada na Equação 5.4. Para encontrar esse limiar, foram testados valores a partir de 0.01 até 0.1. O valor que apresentou melhor desempenho foi 0.046. A medida F1 obtida com esse limiar foi $F1 = 0.555$, calculada utilizando os resultados da Tabela 3 que apresenta a matriz confusão da rede neural. Também é possível visualizar os erros e acertos do modelo na Figura 13, em que 1 significa um acerto na detecção e 0 um erro e a região em vermelho indica o momento em que ocorrem ataques *portscan* e *DDoS* respectivamente. A medida F1 foi escolhida já que é utilizada em problemas binários em que uma das classes é menos comum que a outra [80]. É possível notar na Tabela 3 uma alta taxa de falsos negativos e positivos, mostrando que o modelo ainda precisa ser melhorado.

Tabela 3 – Resultados do limiar F1. Fonte: o próprio autor

	Nenhum Ataque Detectado	Ataque Detectado
Rótulo sem ataque	75921	699
Rótulo com ataque	5747	4033

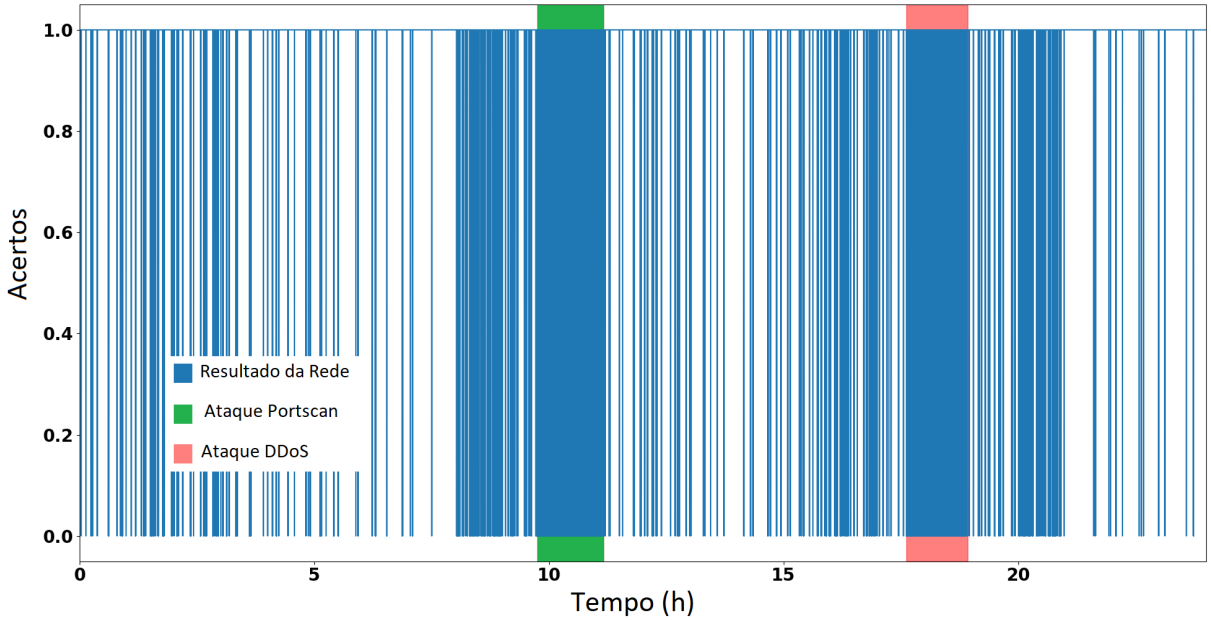

Figura 13 – Resultados do limiar F1. Fonte: O próprio autor.

$$Precisão = \frac{Verdadeiros\ positivos}{Verdadeiros\ positivos + Falsos\ positivos} \tag{5.3}$$

$$Revocação = \frac{Verdadeiros\ positivos}{Verdadeiros\ positivos + Falsos\ negativos} \tag{5.4}$$

$$F1 = 2 * \frac{Precisão * Revocação}{Precisão + Revocação} \tag{5.5}$$

A segunda possibilidade de cálculo de limiar estudada foi a de minimizar a quantidade de falsos positivos (atualmente 699) enquanto mantém a precisão o maior possível. Nesse caso, o limiar 0.076 foi encontrado e seus resultados estão representados na Tabela 4 e Figura 14

Tabela 4 – Resultados do limiar de precisão. Fonte: o próprio autor

	Nenhum Ataque Detectado	Ataque Detectado
Rótulo sem ataque	76620	0
Rótulo com ataque	8597	1183

Analisando a Tabela 4 e a Figura 14, é possível notar uma diminuição nos falsos positivos enquanto há um aumento nos falsos negativos. Isso acontece porque aumentar o limiar o torna mais tolerante a alterações repentinas na rede. Assim, menos variações comuns durante o dia são reconhecidas como ataque ao mesmo tempo que intervalos menos agressivos de ataques não serão detectados.

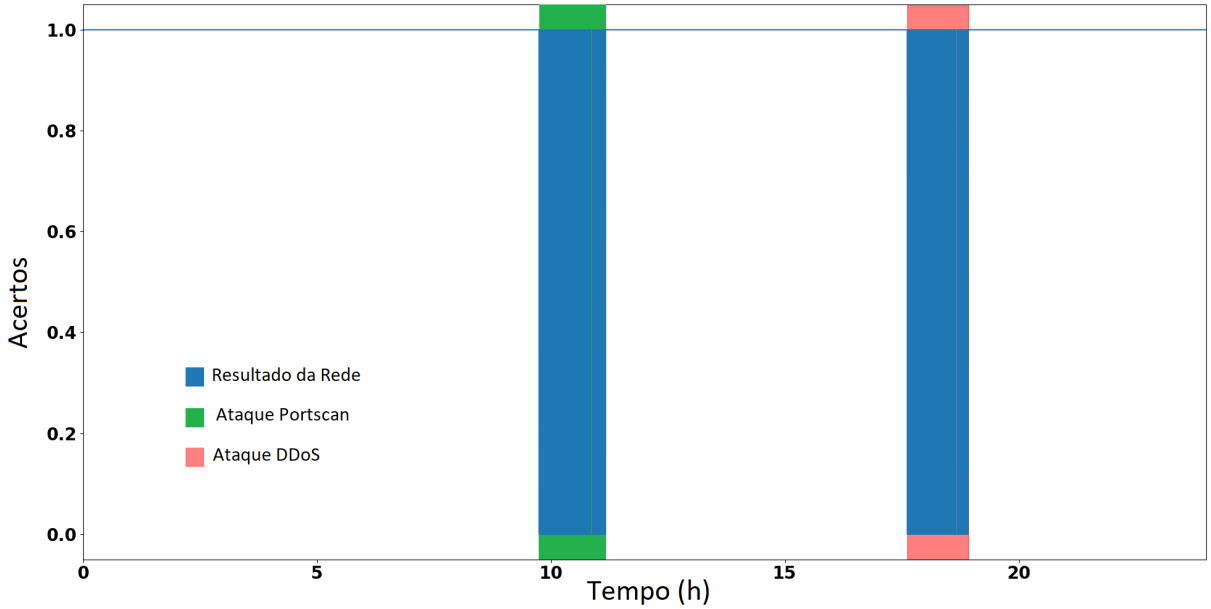

Figura 14 – Resultados do limiar de precisão. Fonte: O próprio autor.

5.4 Um segundo modelo

No trabalho de estágio de Daniel Matheus Brandão Lent, foi utilizada uma metodologia diferente de detecção. A arquitetura da rede é exatamente igual à deste trabalho. A diferença é que a saída da rede não mais irá representar uma dimensão de fluxo, mas sim a probabilidade da ocorrência de uma ataque no segundo analisado.

Para o treinamento da rede neural, um dia com ataque foi utilizado. O rótulo dos dados contém os valores 0 e 1, em que 0 representa a ausência de ataque e 1 a ocorrência de ataque, ambos para o segundo analisado. Assim, como a saída da rede utiliza a função *sigmoid*, o valor de saída estará dentro do intervalo $[0; 1]$. Por isso, um limiar de decisão precisa ser escolhido para decidir sobre a existência de ataque. Para escolher esse valor, foram testados os limiares que maximizavam a medida F1 para o conjunto de treinamento e validação. Cada um dos conjuntos apresentou um intervalo de limiares com o melhor resultado para o conjunto. Esses limiares estão representados na Tabela 5. Pela tabela, é possível verificar que o maior limiar para o dia de treinamento coincide com o menor do dia de validação. Por isso, o valor 72% foi escolhido.

Tabela 5 – Segundo modelo - escolha de limiares. Fonte: o próprio autor

Conjunto de dados	Menor limiar	Maior Limiar
Dia treinamento	67%	72%
Dia validação	72%	94%

Para o conjunto de testes, o quarto dia de dados foi utilizado. A saída da rede foi fixada para 0 caso o valor fosse menor do que 0.72 e para 1 caso o contrário. A Figura 15

mostra o resultado da rede em azul, e destaca em vermelho os horários em que ocorrem ataques. O primeiro ataque é de *portscan* e o segundo *DDoS*. A Tabela 6 apresenta os resultados obtidos durante os testes. É possível notar uma diminuição tanto na taxa de falsos positivos quanto nos falsos negativos em relação ao primeiro modelo.

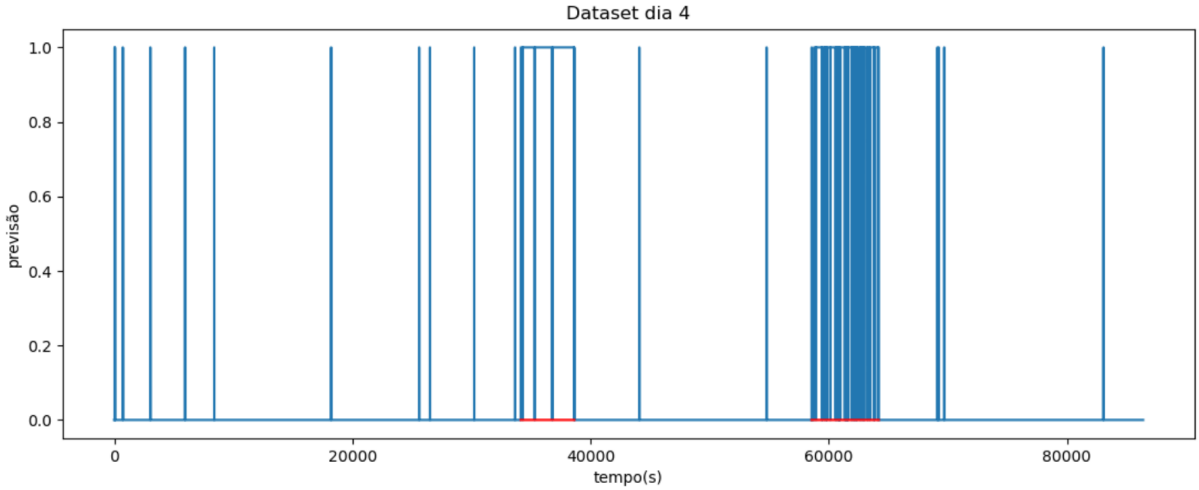

Figura 15 – Resultados do Segundo Modelo. Fonte: O próprio autor.

Tabela 6 – Matriz Confusão do segundo modelo. Fonte: o próprio autor

	Nenhum Ataque Detectado	Ataque Detectado
Rótulo sem ataque	76243	17
Rótulo com ataque	241	9899

6 CONCLUSÃO E TRABALHOS FUTUROS

No trabalho foi desenvolvida uma rede neural convolucional que detecta anomalias em redes *SDN* utilizando 6 dimensões de fluxo, sendo elas *bit* por segundo; pacotes por segundo; entropia de *IP* de origem; entropia de porta de origem; entropia de *IP* de destino e entropia de porta de destino. Também foram estudados dois métodos de escolha de limiares de decisão.

No primeiro método, os resultados mostram uma quantidade significativa de falsos positivos e falsos negativos, resultando em uma medida F1 baixa, mesmo tendo como foco a sua maximização. Esses resultados são o equilíbrio entre falsos positivos e negativos mas tornariam o sistema pouco confiável devido à alta taxa de alarmes falsos.

Por isso, o segundo método visa minimizar os falsos positivos, mesmo que isso seja acompanhado de um aumento nos falsos negativos. Dessa forma, foi possível reduzir os alarmes falsos a quase zero. Os falsos negativos podem não ser tão prejudiciais no caso de ataques *DDoS* pois, em diversos momentos durante os ataques analisados, foram detectadas anomalias, sendo o suficiente para concluir de fato que existe um ataque ocorrendo. O maior defeito desse método é que ele pode levar mais do que 1 segundo para alertar sobre a anomalia já que até o primeiro verdadeiro positivo podem ocorrer diversos falsos negativos, tempo que pode ser significativo para que o ataque cause algum dano. Além disso, anomalias que tenham um impacto menor no tráfego na rede podem passar despercebidas por esse modelo.

Já no modelo do estágio, a taxa de detecção se mostrou superior em relação à primeira metodologia. Contudo, o modelo foi treinado para detectar especificamente os ataques de *DDoS* e *portscan*. Isso pode gerar dificuldades para detectar ataques que sejam diferentes dos já apresentados.

Para trabalhos futuros, pretende-se fazer mais testes com diferentes números de camadas, quantidades de convoluções por camadas, diferentes configurações de camadas densas e até ajustes de parâmetros como preenchimento e passo nas camadas de convolução e *pooling*. Outro ponto estendido, é realizar testes utilizando convoluções de duas dimensões, recebendo mais de um segundo de fluxo como entrada, de forma a oferecer certo "contexto" à rede neural. Além disso, pretende-se fazer testes com novos dados e diferentes métodos de cálculo de limiar e também estudar novas formas de detecção de anomalias.

REFERÊNCIAS

[1] HAMAMOTO, A. H.; CARVALHO, L. F.; SAMPAIO, L. D. H.; ABRãO, T.; PROENçA, M. L. Network anomaly detection system using genetic algorithm and fuzzy logic. *Expert Systems with Applications*, v. 92, p. 390 – 402, 2018. ISSN 0957-4174. Disponível em: <dx.doi.org/10.1016/j.eswa.2017.09.013>.

[2] FERNANDES, G.; RODRIGUES, J. J. P. C.; CARVALHO, L. F.; AL-MUHTADI, J. F.; PROENÇA, M. L. A comprehensive survey on network anomaly detection. *Telecommunication Systems*, v. 70, n. 3, p. 447–489, Mar 2019. ISSN 1572-9451. Disponível em: <dx.doi.org/10.1007/s11235-018-0475-8>.

[3] MASOUDI, R.; GHAFFARI, A. Software defined networks: A survey. *Journal of Network and Computer Applications*, v. 67, p. 1 – 25, 2016. ISSN 1084-8045. Disponível em: <dx.doi.org/10.1016/j.jnca.2016.03.016>.

[4] Farris, I.; Taleb, T.; Khettab, Y.; Song, J. A survey on emerging sdn and nfv security mechanisms for iot systems. *IEEE Communications Surveys Tutorials*, v. 21, n. 1, p. 812–837, 2019. ISSN 1553-877X. Disponível em: <dx.doi.org/10.1109/COMST.2018.2862350>.

[5] NOORIBAKHSH, M.; MOLLAMOTALEBI, M. A review on statistical approaches for anomaly detection in ddos attacks. *Information Security Journal: A Global Perspective*, Taylor Francis, v. 29, n. 3, p. 118–133, 2020. Disponível em: <https://doi.org/10.1080/19393555.2020.1717019>.

[6] Rai, A.; Challa, R. K. Survey on recent ddos mitigation techniques and comparative analysis. In: *2016 Second International Conference on Computational Intelligence Communication Technology (CICT)*. [s.n.], 2016. p. 96–101. ISBN 978-1-5090-0210-8. Disponível em: <dx.doi.org/10.1109/CICT.2016.27>.

[7] AHMED, M.; Naser Mahmood, A.; HU, J. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, v. 60, p. 19 – 31, 2016. ISSN 1084-8045. Disponível em: <dx.doi.org/10.1016/j.jnca.2015.11.016>.

[8] JUNG, J.; KRISHNAMURTHY, B.; RABINOVICH, M. Flash crowds and denial of service attacks: Characterization and implications for cdns and web sites. In: *Proceedings of the 11th International Conference on World Wide Web*. New York, NY, USA: Association for Computing Machinery, 2002. (WWW 02), p. 293304. ISBN 1581134495. Disponível em: <dx.doi.org/10.1145/511446.511485>.

[9] Aksu, D.; Ali Aydin, M. Detecting port scan attempts with comparative analysis of deep learning and support vector machine algorithms. In: *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*. [s.n.], 2018. p. 77–80. Disponível em: <10.1109/IBIGDELFT.2018.8625370>.

[10] Gadge, J.; Patil, A. A. Port scan detection. In: *2008 16th IEEE International Conference on Networks*. [s.n.], 2008. p. 1–6. Disponível em: <10.1109/ICON.2008.4772622>.

[11] AMANULLAH, M. A.; HABEEB, R. A. A.; NASARUDDIN, F. H.; GANI, A.; AHMED, E.; NAINAR, A. S. M.; AKIM, N. M.; IMRAN, M. Deep learning and big data technologies for iot security. *Computer Communications*, v. 151, p. 495 – 517, 2020. ISSN 0140-3664. Disponível em: <https://dx.doi.org/10.1016/j.comcom.2020.01.016>.

[12] Kolias, C.; Kambourakis, G.; Stavrou, A.; Voas, J. Ddos in the iot: Mirai and other botnets. *Computer*, v. 50, n. 7, p. 80–84, 2017. Disponível em: <dx.doi.org/10.1109/MC.2017.201>.

[13] JOSE, S.; MALATHI, D.; REDDY, B.; JAYASEELI, D. A survey on anomaly based host intrusion detection system. *Journal of Physics: Conference Series*, IOP Publishing, v. 1000, p. 012049, apr 2018. Disponível em: <dx.doi.org/10.1088/1742-6596/1000/1/012049>.

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

[15] 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.n.], 2013. p. 735–740. Disponível em: <dx.doi.org/10.1109/GLOCOM.2013.6831160>.

[16] Carvalho, L. F.; Fernandes, G.; Rodrigues, J. J. P. C.; Mendes, L. S.; Proença, M. L. A novel anomaly detection system to assist network management in sdn environment. In: *2017 IEEE International Conference on Communications (ICC)*. [s.n.], 2017. Disponível em: <dx.doi.org/10.1109/ICC.2017.7997214>.

[17] BURKOV, A. *The Hundred-Page Machine Learning Book*. 1. ed. Kindle Direct Publishing, 2019. ISBN 9781790485000. Disponível em: <dx.doi.org/10.1080/15228053.2020.1766224>.

[18] SCHMIDHUBER, J. Deep learning in neural networks: An overview. *Neural networks : the official journal of the International Neural Network Society*, v. 61, 2015. Disponível em: <dx.doi.org/10.1016/j.neunet.2014.09.003>.

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

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

[21] O'SHEA, K.; NASH, R. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015. Disponível em: <http://arxiv.org/abs/1511.08458>.

[22] Haider, S.; Akhunzada, A.; Ahmed, G.; Raza, M. Deep learning based ensemble convolutional neural network solution for distributed denial of service detection in sdns. In: *2019 UK/ China Emerging Technologies (UCET)*. [s.n.], 2019. p. 1–4. Disponível em: <dx.doi.org/10.1109/UCET.2019.8881856>.

[23] BENGIO, Y. Learning deep architectures for ai. *Foundations and Trendső in Machine Learning*, v. 2, n. 1, p. 1–127, 2009. ISSN 1935-8237. Disponível em: <10.1561/2200000006>.

[24] Kreutz, D.; Ramos, F. M. V.; Veríssimo, P. E.; Rothenberg, C. E.; Azodolmolky, S.; Uhlig, S. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, v. 103, n. 1, p. 14–76, 2015. Disponível em: <10.1109/JPROC.2014.2371999>.

[25] Scaranti, G. F.; Carvalho, L. F.; Barbon, S.; Proença, M. L. Artificial immune systems and fuzzy logic to detect flooding attacks in software-defined networks. *IEEE Access*, v. 8, p. 100172–100184, 2020. Disponível em: <dx.doi.org/10.1109/ACCESS.2020.2997939>.

[26] LAZAREVIC, A.; ERTOZ, L.; KUMAR, V.; OZGUR, A.; SRIVASTAVA, J. A comparative study of anomaly detection schemes in network intrusion detection. In: _____. *Proceedings of the 2003 SIAM International Conference on Data Mining*. [s.n.], 2003. p. 25–36. ISBN 978-1-61197-273-3. Disponível em: <dx.doi.org/10.1137/1.9781611972733.3>.

[27] Yin, C.; Zhu, Y.; Fei, J.; He, X. A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access*, v. 5, p. 21954–21961, 2017. ISSN 2169-3536. Disponível em: <dx.doi.org/10.1109/ACCESS.2017.2762418>.

[28] Shone, N.; Ngoc, T. N.; Phai, V. D.; Shi, Q. A deep learning approach to network intrusion detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, v. 2, n. 1, p. 41–50, 2018. ISSN 2471-285X. Disponível em: <dx.doi.org/10.1109/TETCI.2017.2772792>.

[29] UMER, M. F.; SHER, M.; BI, Y. Flow-based intrusion detection: Techniques and challenges. *Computers Security*, v. 70, p. 238 – 254, 2017. ISSN 0167-4048. Disponível em: <dx.doi.org/10.1016/j.cose.2017.05.009>.

[30] OH, D.; KIM, D.; RO, W. W. A malicious pattern detection engine for embedded security systems in the internet of things. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 14, n. 12, p. 24188–24211, 2014.

[31] PENA, E. H.; CARVALHO, L. F.; Barbon Jr., S.; RODRIGUES, J. J.; Proença Jr., M. L. Anomaly detection using the correlational paraconsistent machine with digital signatures of network segment. *Information Sciences*, v. 420, p. 313 – 328, 2017. ISSN 0020-0255. Disponível em: <dx.doi.org/10.1016/j.ins.2017.08.074>.

[32] OSANAIYE, O.; CHOO, K.-K. R.; DLODLO, M. Distributed denial of service (ddos) resilience in cloud: Review and conceptual cloud ddos mitigation framework. *Journal of Network and Computer Applications*, v. 67, p. 147 – 165, 2016. ISSN 1084-8045. Disponível em: <dx.doi.org/10.1016/j.jnca.2016.01.001>.

[33] de Assis, M. V.; CARVALHO, L. F.; RODRIGUES, J. J.; LLORET, J.; Proença Jr, M. L. Near real-time security system applied to sdn environments in iot networks using convolutional neural network. *Computers Electrical Engineering*, v. 86, p. 106738, 2020. ISSN 0045-7906. Disponível em: <dx.doi.org/10.1016/j.compeleceng.2020.106738>.

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

[35] RING, M.; WUNDERLICH, S.; SCHEURING, D.; LANDES, D.; HOTHO, A. A survey of network-based intrusion detection data sets. *Computers Security*, v. 86, p. 147 – 167, 2019. ISSN 0167-4048. Disponível em: <dx.doi.org/10.1016/j.cose.2019.06.005>.

[36] CARVALHO, L. F.; BARBON, S.; MENDES, L. de S.; PROENçA, M. L. Unsupervised learning clustering and self-organized agents applied to help network management. *Expert Systems with Applications*, v. 54, p. 29 – 47, 2016. ISSN 0957-4174. Disponível em: <dx.doi.org/10.1016/j.eswa.2016.01.032>.

[37] de Assis, M. V. O.; Carvalho, L. F.; Rodrigues, J. J. P. C.; Proença, M. L. Holt-winters statistical forecasting and aco metaheuristic for traffic characterization. In: *2013 IEEE International Conference on Communications (ICC)*. [s.n.], 2013. p. 2524–2528. Disponível em: <dx.doi.org/10.1109/ICC.2013.6654913>.

[38] Novaes, M. P.; Carvalho, L. F.; Lloret, J.; Proença, M. L. Long short-term memory and fuzzy logic for anomaly detection and mitigation in software-defined network environment. *IEEE Access*, v. 8, p. 83765–83781, 2020. Disponível em: <dx.doi.org/10.1109/ACCESS.2020.2992044>.

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

[40] RAJESH, S. Protection from application layer ddos attacks for popular websites. *International Journal of Computer and Electrical Engineering*, v. 15, n. 6, p. 555–558, 2013. ISSN 1793-8163. Disponível em: <dx.doi.org/10.7763/IJCEE.2013.V5.771>.

[41] Buczak, A. L.; Guven, E. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys Tutorials*, v. 18, n. 2, p. 1153–1176, 2016. ISSN 1553-877X. Disponível em: <dx.doi.org/10.1109/COMST.2015.2494502>.

[42] KOUROU, K.; EXARCHOS, T. P.; EXARCHOS, K. P.; KARAMOUZIS, M. V.; FOTIADIS, D. I. Machine learning applications in cancer prognosis and prediction. *Computational and Structural Biotechnology Journal*, v. 13, p. 8 – 17, 2015. ISSN 2001-0370. Disponível em: <dx.doi.org/10.1016/j.csbj.2014.11.005>.

[43] LAI, M. Giraffe: Using deep reinforcement learning to play chess. *arXiv preprint arXiv:1509.01549*, 2015.

[44] MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. *Foundations of Machine Learning*. [S.l.]: The MIT Press, 2012. ISBN 026201825X.

[45] Keller, J. M.; Liu, D.; Fogel, D. B. *Fundamentals of Computational Intelligence: Neural Networks, Fuzzy Systems, and Evolutionary Computation*. IEEE Press, 2016. ISBN 9781119214403. Disponível em: <https://ieeexplore.ieee.org/servlet/opac?bknumber=7547467>.

[46] K., V.; K., S. Towards activation function search for long short-term model network: A differential evolution based approach. *Journal of King Saud University - Computer and Information Sciences*, 2020. ISSN 1319-1578. Disponível em: <dx.doi.org/10.1016/j.jksuci.2020.04.015>.

[47] SHARMA, S.; SHARMA, S. Activation functions in neural networks. *Journal of Engineering Applied Sciences and Technology*, v. 4, n. 12, p. 310–316, 2020. ISSN 2455-2143. Disponível em: <https://www.ijeast.com/papers/310-316,Tesma412,IJEAST.pdf>.

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

[49] MAHDAVIFAR, S.; GHORBANI, A. A. Application of deep learning to cybersecurity: A survey. *Neurocomputing*, v. 347, p. 149 – 176, 2019. ISSN 0925-2312. Disponível em: <dx.doi.org/10.1016/j.neucom.2019.02.056>.

[50] HEWAMALAGE, H.; BERGMEIR, C.; BANDARA, K. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, Elsevier BV, Aug 2020. ISSN 0169-2070. Disponível em: <dx.doi.org/10.1016/j.ijforecast.2020.06.008>.

[51] HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Computation*, v. 9, n. 8, p. 1735–1780, 1997. Disponível em: <dx.doi.org/10.1162/neco.1997.9.8.1735>.

[52] GRAVES, A.; SCHMIDHUBER, J. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, v. 18, n. 5, p. 602 – 610, 2005. ISSN 0893-6080. IJCNN 2005. Disponível em: <dx.doi.org/10.1016/j.neunet.2005.06.042>.

[53] ECK, D.; SCHMIDHUBER, J. Learning the long-term structure of the blues. In: DORRONSORO, J. R. (Ed.). *Artificial Neural Networks — ICANN 2002*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002. p. 284–289. ISBN 978-3-540-46084-8. Disponível em: <dx.doi.org/10.1007/3-540-46084-5_47>.

[54] OLAH, C. *Understanding LSTM Networks*. [S.l.], Acessado: 2020–12–10. Disponível em: <https://colah.github.io/posts/2015-08-Understanding-LSTMs>.

[55] BROWNLEE, J. *Develop Sequence Prediction Models With Deep Learning*. [s.n.], 2017. Disponível em: <https://machinelearningmastery.com/lstms-with-python/>.

[56] Creswell, A.; White, T.; Dumoulin, V.; Arulkumaran, K.; Sengupta, B.; Bharath, A. A. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, v. 35, n. 1, p. 53–65, 2018. Disponível em: <dx.doi.org/10.1109/MSP.2017.2765202>.

[57] ALQAHTANI, H.; KAVAKLI-THORNE, M.; KUMAR, G. Applications of generative adversarial networks (gans): An updated review. *Archives of Computational Methods in Engineering*, Dec 2019. ISSN 1886-1784. Disponível em: <dx.doi.org/10.1007/s11831-019-09388-y>.

[58] ZHANG, G.; LIU, Y.; JIN, X. A survey of autoencoder-based recommender systems. *Frontiers of Computer Science*, Springer, p. 1–21, 2020. ISSN 2095-2236. Disponível em: <dx.doi.org/10.1007/s11704-018-8052-6>.

[59] LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, v. 521, n. 7553, p. 436–444, May 2015. ISSN 1476-4687. Disponível em: <dx.doi.org/10.1038/nature14539>.

[60] DRUZHKOV, P. N.; KUSTIKOVA, V. D. A survey of deep learning methods and software tools for image classification and object detection. *Pattern Recognition and Image Analysis*, v. 26, n. 1, p. 9–15, Jan 2016. ISSN 1555-6212. Disponível em: <dx.doi.org/10.1134/S1054661816010065>.

[61] KHAN, A.; SOHAIL, A.; ZAHOORA, U.; QURESHI, A. S. A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, v. 53, n. 8, p. 5455–5516, Dec 2020. ISSN 1573-7462. Disponível em: <dx.doi.org/10.1007/s10462-020-09825-6>.

[62] Cokun, M.; Uçar, A.; Yildirim, .; Demir, Y. Face recognition based on convolutional neural network. In: *2017 International Conference on Modern Electrical and Energy Systems (MEES)*. [s.n.], 2017. p. 376–379. Disponível em: <dx.doi.org/10.1109/MEES.2017.8248937>.

[63] MANKAR, V.; BHELE, S. A review paper on face recognition techniques. *International Journal of Advanced Research in Computer Engineering Technology*, v. 1, p. 339–346, 10 2012. ISSN 2494-9150. Disponível em: <https://www.ijream.org/papers/IJREAMV02I01876.pdf>.

[64] PARKHI, O. M.; VEDALDI, A.; ZISSERMAN, A. Deep face recognition. In: XIE, X.; JONES, M. W.; TAM, G. K. L. (Ed.). *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press, 2015. p. 41.1–41.12. ISBN 1-901725-53-7. Disponível em: <dx.doi.org/10.5244/C.29.41>.

[65] Young, T.; Hazarika, D.; Poria, S.; Cambria, E. Recent trends in deep learning based natural language processing [review article]. *IEEE Computational Intelligence Magazine*, v. 13, n. 3, p. 55–75, 2018. Disponível em: <dx.doi.org/10.1109/MCI.2018.2840738>.

[66] LI, H. Deep learning for natural language processing: advantages and challenges. *National Science Review*, v. 5, n. 1, p. 24–26, 09 2017. ISSN 2095-5138. Disponível em: <dx.doi.org/10.1093/nsr/nwx110>.

[67] YIN, W.; KANN, K.; YU, M.; SCHÜTZE, H. Comparative study of CNN and RNN for natural language processing. *CoRR*, abs/1702.01923, 2017. Disponível em: <http://arxiv.org/abs/1702.01923>.

[68] CONNEAU, A.; SCHWENK, H.; BARRAULT, L.; LECUN, Y. Very deep convolutional networks for text classification. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. [s.n.], 2017. v. 1, p. 1107–1116. Disponível em: <dx.doi.org/10.18653/v1/E17-1104>.

[69] Kwon, D.; Natarajan, K.; Suh, S. C.; Kim, H.; Kim, J. An empirical study on network anomaly detection using convolutional neural networks. In: *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. [s.n.], 2018. p. 1595–1598. Disponível em: <dx.doi.org/10.1109/ICDCS.2018.00178>.

[70] PANIGRAHI, R.; BORAH, S. A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems. *International Journal of Engineering & Technology*, v. 7, p. 479–482, 01 2018. ISSN 2227-524X.

[71] 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.n.], 2005. p. 290–295. Disponível em: <dx.doi.org/10.1109/AICT.2005.26>.

[72] Dromard, J.; Roudière, G.; Owezarski, P. Online and scalable unsupervised network anomaly detection method. *IEEE Transactions on Network and Service Management*, v. 14, n. 1, p. 34–47, 2017. Disponível em: <dx.doi.org/10.1109/TNSM.2016.2627340>.

[73] Aygun, R. C.; Yavuz, A. G. Network anomaly detection with stochastically improved autoencoder based models. In: *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*. [s.n.], 2017. p. 193–198. Disponível em: <dx.doi.org/10.1109/CSCloud.2017.39>.

[74] Naseer, S.; Saleem, Y.; Khalid, S.; Bashir, M. K.; Han, J.; Iqbal, M. M.; Han, K. Enhanced network anomaly detection based on deep neural networks. *IEEE Access*, v. 6, p. 48231–48246, 2018. Disponível em: <dx.doi.org/10.1109/ACCESS.2018.2863036>.

[75] BEREZISKI, P.; JASIUL, B.; SZPYRKA, M. An entropy-based network anomaly detection method. *Entropy*, v. 17, n. 4, p. 2367–2408, 2015. ISSN 1099-4300. Disponível em: <dx.doi.org/10.3390/e17042367>.

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

[77] Qin, Y.; Wei, J.; Yang, W. Deep learning based anomaly detection scheme in software-defined networking. In: *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. [s.n.], 2019. p. 1–4. Disponível em: <dx.doi.org/10.23919/APNOMS.2019.8892873>.

[78] WEHRL, A. General properties of entropy. *Rev. Mod. Phys.*, American Physical Society, v. 50, p. 221–260, Apr 1978. Disponível em: <dx.doi.org/10.1103/RevModPhys.50.221>.

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

[80] LIPTON, Z. C.; ELKAN, C.; NARYANASWAMY, B. Optimal thresholding of classifiers to maximize f1 measure. In: CALDERS, T.; ESPOSITO, F.; HÜLLERMEIER, E.; MEO, R. (Ed.). *Machine Learning and Knowledge Discovery in Databases*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. p. 225–239. ISBN 978-3-662-44851-9. Disponível em: <dx.doi.org/10.1007/978-3-662-44851-9_15>.