Universidade Estadual de Londrina

VINICIUS MARINO LUCIANO

DETECÇÃO DE ATAQUES EM INTERNET DAS COISAS ATRAVÉS DE TÉCNICAS DE APRENDIZADO PROFUNDO

LONDRINA
2022

VINICIUS MARINO LUCIANO

DETECÇÃO DE ATAQUES EM INTERNET DAS COISAS ATRAVÉS DE TÉCNICAS DE APRENDIZADO PROFUNDO

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. Bruno Bogaz Zarpelão

LONDRINA
2022

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

Luciano, Vinicius.
Detecção de ataques em Internet das Coisas através de técnicas de Aprendizado Profundo / Vinicius Luciano. - Londrina, 2022.
48 f.

Orientador: Bruno Zarpelão.
Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Universidade Estadual de Londrina, Centro de Ciências Exatas, Graduação em Ciência da Computação, 2022.
Inclui bibliografia.

1. Internet das Coisas - TCC. 2. Aprendizado Profundo - TCC. 3. Sistemas de detecção de intrusão - TCC. 4. Detecção de Anomalias - TCC. I. Zarpelão, Bruno. II. Universidade Estadual de Londrina. Centro de Ciências Exatas. Graduação em Ciência da Computação. III. Título.

CDU 519

VINICIUS MARINO LUCIANO

DETECÇÃO DE ATAQUES EM INTERNET DAS COISAS ATRAVÉS DE TÉCNICAS DE APRENDIZADO PROFUNDO

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. Bruno Bogaz Zarpelão
Universidade Estadual de Londrina

Prof. Dr. Adilson Luiz Bonifácio
Universidade Estadual de Londrina – UEL

Prof. Gabriel Keith Tazima
Universidade Estadual de Londrina – UEL

Londrina, 02 de junho de 2022.

AGRADECIMENTOS

Primeiramente agradeço a todas as pessoas que passaram pela minha vida, que me fizeram me tornar quem eu sou hoje. Gostaria de agradecer aos meus pais, que por mais distante que tiveram, sempre confiaram em mim. Agradecimento especial ao Prof. Dr. Bruno Bogaz Zarpelão por todos os conhecimentos, e principalmente por entender, apoiar e respeitar o meu ritmo de desenvolvimento.

Por fim gostaria de agradecer aos meus colegas de turma, em especial aos meus amigos que me ensinaram tanto durante essa jornada, Matheus e Melvi.

LUCIANO, V. M.. **Detecção de ataques em Internet das Coisas através de técnicas de aprendizado profundo**. 2022. 50f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2022.

RESUMO

A Internet das Coisas (IoT - Internet of Things) é uma tendência global em evolução e está cada vez mais difundida. Apesar do seu potencial, a propagação de dispositivos IoT que possuem pouca segurança incorporada tem trazido grandes desafios para a segurança das redes. Portanto, muitas propostas de sistemas de detecção de intrusão têm surgido a fim de mitigar este problema. No entanto, grande parte destes trabalhos utilizam-se de algoritmos supervisionados, o que torna-se um problema, pois a rotulação dos dados de treinamento é uma tarefa demorada e custosa. Como solução para este problema, este trabalho propõe um sistema de detecção de intrusão que utiliza um Autoencoder como técnica de aprendizado profundo One-Class, com o objetivo de treinamento de um modelo com alta capacidade de aprendizado sem a necessidade de rotulação dos dados. Experimentos realizados com conjuntos de dados públicos contendo diversos cenários de ataque a dispositivos IoT reais mostraram que o sistema proposto foi capaz de identificar os ataques com alta precisão e com baixa taxa de falsos positivos para a maioria dos casos.

Palavras-chave: Aprendizado Profundo. Detecção de Anomalias. Sistemas de Detecção de Intrusão. Internet das Coisas.

LUCIANO, V. M.. **Attack detection in the Internet of Things through deep learning techniques**. 2022. 50p. Final Project (Bachelor of Science in Computer Science) – State University of Londrina, Londrina, 2022.

ABSTRACT

The Internet of Things (IoT) is a global trend that is evolving and settling. Despite its potential, the wide presence of IoT devices with low security brings challenges to network security. Therefore, many proposals for intrusion detection systems have emerged to mitigate this problem. However, most of these studies use supervised algorithms, which becomes a problem, as labeling the training data is a slow and difficult task. As a solution to this problem, this work proposes an intrusion detection system through the use of an Autoencoder as a One-Class deep learning technique, aiming to train a model with high learning capacity without the need for data labeling. Experiments performed with public datasets containing multiple attack scenarios on real IoT devices showed that the proposed system was able to identify attacks with high accuracy and low false positive rate for most cases.

Keywords: Deep Learning. Anomaly Detection. Intrusion Detection Systems. Internet of Things.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O crescimento do número de dispositivos de Internet das Coisas (*Internet of Things* - IoT) conectados à Internet trouxe diversos desafios para a segurança das redes. Apesar de gerar e manipular muitas informações privadas, esses dispositivos, normalmente devido ao baixo custo, possuem pouca ou nenhuma segurança incorporada [1].

Devido ao fato desses dispositivos não serem seguros como os computadores, mas ainda assim se envolverem em tarefas sensíveis à segurança e privacidade dos usuários, a IoT tem sido um alvo recorrente para pessoas mal intencionadas [2], que podem utilizar desses equipamentos para extrair informações confidenciais e também direcionar ataques de negação de serviço, que tem como objetivo tornar os recursos de um sistema indisponíveis através de uma rede de *bots* em larga escala [3].

Com essas constantes ameaças, a criação de sistemas de detecção de intrusão (*Intrusion Detection System* - IDS) tem sido fundamental para esses dispositivos [4]. Estes sistemas são uma das principais ferramentas quando se trata de segurança das redes tradicionais. No entanto, para dispositivos IoT ainda se mantém presentes alguns problemas em aberto devido às características particulares desta tecnologia, como recursos limitados, protocolos de comunicação específicos e comportamentos padronizados [5, 6].

Portanto, o desenvolvimento de sistemas de detecção de intrusão para IoT tem sido um grande desafio para pesquisadores de segurança da informação e administradores de redes. Pensando nisso, muitos trabalhos surgiram com o objetivo de mitigar essas ameaças. No entanto, muitos destes trabalhos se baseiam em técnicas de aprendizado supervisionado, o que dificulta o treinamento do algoritmo, bem como a adaptação a novos ataques [7, 8, 9].

Por outro lado, algoritmos semi-supervisionados *One-Class* necessitam modelar apenas um único padrão e utilizar este para discernir se uma nova amostra pertence ou não ao padrão, ou seja, se é ou não um ataque. Isso leva a uma facilidade considerável de treinamento e retreinamento do algoritmo, uma vez que classificadores *Multi-Class* necessitam de amostras de todas as classes catalogadas no treinamento, algo que demanda um considerável esforço [1, 2, 10].

Pensando nisso, a proposta deste trabalho é criar um sistema de detecção de intrusão baseado em rede através de técnicas de aprendizado profundo *One-Class Classification*. A proposta é uma abordagem baseada em *deep learning*, que utiliza um Autoencoder, rede neural artificial usada para aprender codificações eficientes de dados não rotulados, para realizar a detecção dos ataques.

Para o treinamento do algoritmo, primeiramente é extraído um conjunto de características do tráfego normal dos dispositivos conectados na rede. Este conjunto de características será então utilizado para que o Autoencoder aprenda o comportamento normal destes dispositivos. Portanto será utilizado um Autoencoder para cada dispositivo conectado, uma vez que os comportamentos normais se diferem entre diferentes equipamentos. Quando novas amostras são adicionadas, o Autoencoder tenta reconstruir suas características, e quando ele falha, tem-se um forte indício de que a amostra observada é anômala.

A utilização de um Autoencoder para essa abordagem traz diversas vantagens. Dentre elas, a sua excelente capacidade na aprendizagem de dados complexos se destaca, podendo reduzir consideravelmente o número de falsos positivos quando comparado com outros algoritmos comumente utilizados com o mesmo objetivo [11]. Além disso, devido ao fato dos dispositivos IoT, normalmente, não possuírem uma vasta variedade de funções, utilizar o Autoencoder para aprender o comportamento padrão certamente se torna uma tarefa simplificada, facilitando também a detecção de um ataque quando o comportamento se difere consideravelmente do padrão aprendido.

No Capítulo 2, é apresentada a fundamentação teórica e o estado da arte atual, trazendo definições de Internet das Coisas e os principais ataques relacionados, sistemas de detecção de intrusão e Aprendizado de Máquina, além de mostrar os principais trabalhos relacionados. No Capítulo 3 é apresentada a solução proposta no trabalho, juntamente com as características extraídas que foram utilizadas nos experimentos. No Capítulo 4 são apresentados os experimentos realizados, assim como seus resultados. Por fim, no Capítulo 5 é discutida a conclusão do projeto e apresentado possíveis propostas para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA E ESTADO DA ARTE

2.1 Internet das Coisas

A Internet das Coisas (do inglês *Internet of Things*) pode ser descrita como uma rede de dispositivos físicos conectados entre si através de sensores, circuitos e softwares que tem como objetivo permitir a coleta e troca de dados entre eles [12].

Nos últimos anos, a Internet das Coisas vem em uma crescente considerável no cenário global da tecnologia. Estes equipamentos estão presentes em diversas áreas, como: transporte, infraestrutura civil, saúde, agricultura, indústrias, uso doméstico, etc [2].

Apesar de todas as facilidades que esses equipamentos agregam à vida humana, uma única vulnerabilidade em tais sistemas pode levar a consequências que vão desde a perda de privacidade, danos físicos, financeiros e até mesmo a possibilidade de colocar vidas em risco [13]. Portanto, a segurança desses dispositivos tem sido um grande desafio para a comunidade científica, tendo em vista que, para esses equipamentos, métodos convencionais de segurança podem ser inadequados [6].

Diferentes razões explicam este problema. Primeiramente, a arquitetura de rede desses dispositivos se difere das redes tradicionais. Em redes tradicionais, os sistemas finais se conectam diretamente com nós específicos (por exemplo, roteadores e pontos de acesso sem fio) que se responsabilizam por transmitir os pacotes ao destino. Por outro lado, em redes IoT, ao mesmo tempo que os dispositivos trabalham como sistemas finais, eles podem também serem responsáveis por realizar o encaminhamento dos pacotes, sem que haja a conexão com um nó específico para isso, alterando os fluxos dos pacotes na rede.

Além disso, esses dispositivos costumam usar protocolos de rede específicos, que não são utilizados em redes tradicionais, como o *IEEE 802.15.4*, *IPv6 over Low-power Wireless Personal Area Network (6LoWPAN)*, *IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL)* e *Constrained Application Protocol (CoAP)* [14, 15, 6]. Com o uso de diferentes protocolos, diferentes vulnerabilidades são apresentadas, dificultando a utilização de sistemas de detecção de intrusão convencionais.

Por fim, a segurança desses equipamentos se agrava devido ao fato de que a maioria dos fabricantes não trabalham com questões relacionadas à segurança.

2.1.1 Ataques em Internet das Coisas

Atualmente existem diversos tipos de ataques e ameaças em Internet das Coisas. Dentre os principais tipos, existem aqueles com foco em *hardware*, como etiquetas RFID

(*Radio Frequency Identification*), microcontroladores e sensores. Há também os ataques focados em descobrir potenciais falhas em protocolos IoT, como roteamento, conectividade e protocolos de comunicação. Além disso, existem os ataques baseados em dados, sejam eles armazenados nos dispositivos IoT ou em nuvem. Por fim, os ataques em *software* incluindo aplicações contidas nos dispositivos ou na nuvem, como *firmwares*, sistemas operacionais e serviços [16, 17].

Dentro destas classificações existem ataques como *Hardware Trojan*, que tem como principal propósito modificar o circuito integrado do dispositivo a fim de obter acesso a dados confidenciais. *Tag cloning*, que consiste na captura de tags e construção de réplicas para comprometer sistemas RFID. *Spoofing*, onde o atacante utiliza de uma falsa identidade, como endereço MAC (*Media Access Control*) ou IP (*Internet Protocol*) de um usuário legítimo para obter acesso a dados confidenciais. *Man in the middle (MITM)*, no qual o invasor intercepta a comunicação entre dois sistemas se passando por um sistema legítimo, podendo receber informações confidenciais e enviar mensagens maliciosas. *Denial of service (DoS)*, que consiste em uma tentativa de tornar algum recurso indisponível através de uma larga escala de solicitações ao dispositivo, podendo esgotá-lo em processamento ou energia. *TCP-UDP Port scan*, que tem como objetivo realizar uma varredura de portas a fim de descobrir pontos de falha. Além destes, muitos outros ataques podem ser observados em Abdul et al. [16].

2.1.1.1 Botnets

Botnets são redes de dispositivos infectados por *softwares* maliciosos que podem ser controladas remotamente por agentes mal intencionados. Tendo o controle de diversos dispositivos, as *botnets* são, geralmente, utilizadas para enviar *spam*, vírus, roubar dados pessoais ou até mesmo executar ataques DDoS [18].

Uma *botnet* é composta basicamente por três pilares: os *bots*, que são os dispositivos infectados que compõem a rede; o *botmaster* que, é o usuário mal intencionado que controla todos estes dispositivos; e a estrutura de Comando e Controle (*Command and Control* - C&C), que por sua vez é responsável por fazer a comunicação entre o *botmaster* e os *bots*. Por meio da estrutura C&C, o *botmaster* pode enviar comandos para a rede de *bots* e também receber informações vindas dela [19].

A infecção dos dispositivos pode ser feita de diversas maneiras, seja por meio de cavalos de troia, engenharia social ou até mesmo obtendo acesso através de senhas padrão, vulnerabilidade comum em câmeras de segurança e roteadores domésticos.

Já a estrutura de Comando e Controle pode ser construída de diferentes formas. A primeira e mais simples é através do modelo cliente-servidor, no qual uma única máquina é responsável por enviar os comandos de ataque para todos os nós da rede. Neste tipo de estrutura, basta desligar a máquina *botmaster* para desligar toda a *botnet*. O segundo mo-

delo é conhecido como "*Peer-to-Peer*", no qual cada nó infectado comunica-se diretamente com alguns outros na rede, sendo ela inteiramente conectada. Dessa forma, remover um dispositivo ou outro não interrompe o funcionamento da *botnet*.

O ataque por sua vez é realizado a partir do momento em que o *botmaster* envia o comando para os *bots* através da estrutura de Comando e Controle. A partir desse ponto, todos os *bots* recebem a informação e inicia-se a execução do comando recebido.

Dentre as diversas formas de ataque, as *botnets* têm se destacado como uma grande ameaça à segurança cibernética [20], principalmente quando se trata das *botnets* baseadas em dispositivos IoT. Isso se deve principalmente à grande proliferação destes dispositivos, que normalmente são mais facilmente infectados que os computadores desktop [21]. Uma das *botnets* mais marcantes foi a Mirai. Composta principalmente por dispositivos embarcados e IoT, foi responsável por ataques DDoS de grande escala ao reunir um grande volume de dispositivos [22].

Apesar de existirem diversos trabalhos com objetivo de identificar e mitigar esse problema, a maior parte da literatura apresenta soluções que conseguem detectar a *botnet* durante um ataque em andamento, ou seja, após ela já ter recebido o comando do *botmaster* e estar afetando o alvo. No entanto, seria melhor que as soluções detectassem as *botnets* no momento em que nós estão sendo infectados e ainda não começaram a atacar [2, 23].

2.2 Sistemas de Detecção de Intrusão

Sistemas de Detecção de Intrusão (do inglês *Intrusion Detection System* - IDS) podem ser definidos como um *hardware*, ou um *software* ou uma combinação de ambos com o objetivo de monitorar uma rede a fim de encontrar eventos que possam violar suas regras de segurança [24, 25].

Com o grande aumento das ameaças contra as redes e a dificuldade que se tem para mitigação dessas ameaças, os Sistemas de Detecção de Intrusão têm recebido atenção de grande parte dos pesquisadores [26]. A principal função do sistema é detectar uma atividade suspeita e posteriormente relatar alertas aos administradores da rede em tempo hábil.

Há alguns anos que os IDSs vêm sendo uma ferramenta de reconhecida importância na proteção, alerta e monitoramento de redes. Entretanto, a utilização de técnicas tradicionais associadas a dispositivos IoT não tem demonstrado bons resultados, devido ao fato destes dispositivos possuírem particularidades que os diferem de dispositivos convencionais, como recursos limitados, protocolos e arquitetura específicos [6].

Os IDSs podem ser classificados de duas diferentes formas: *Host-based Intrusion*

Detection Systems (HIDS) e *Network-based Intrusion Detection Systems (NIDS)*. Os *Host-based IDSs* são instalados no próprio dispositivo sob monitoramento a fim de avaliar tanto os tráfegos na rede, como também recursos como uso de CPU, memória, energia, entre outros. Já os *Network-based IDSs*, normalmente, tem como objetivo monitorar apenas o tráfego de rede a fim de encontrar atividades maliciosas [27, 6].

Além disso, os IDSs podem ser implementados basicamente de duas diferentes formas. A primeira, conhecida como *signature-based IDS*, quando o sistema deve conhecer previamente os padrões utilizados em cada tipo de ataque que pode ser identificado. A vantagem nesse caso é a rápida identificação quando um ataque conhecido ocorre. Sua desvantagem é justamente quando ocorre um ataque desconhecido. Caso os padrões deste ataque não sejam similares a algum ataque conhecido pelo sistema, o tráfego poderá ser classificado como normal [27, 28].

A segunda abordagem é conhecida como *anomaly-based IDS*. Neste caso, o sistema irá traçar os padrões e comportamentos normais no dispositivo ou tráfego de rede. Quando um comportamento que difere desses padrões ocorre, ele é catalogado como anômalo. A grande vantagem dessa abordagem é que o sistema não necessita conhecer previamente os padrões dos ataques. A desvantagem é que esta abordagem leva normalmente a uma quantidade considerável de falsos positivos [2, 28].

2.3 Aprendizado de Máquina

Aprendizado de Máquina (do inglês *Machine Learning*) é uma subárea da Inteligência Artificial que tem como objetivo principal o desenvolvimento de técnicas computacionais com o intuito de encontrar padrões que possam utilizados para tomar decisões com base em experiências acumuladas durante o treinamento [29].

Existem basicamente dois tipos de problemas que são resolvidos por meio do aprendizado de máquina: regressão e classificação. Os problemas de regressão podem ser definidos resumidamente como uma predição de valores contínuos, por exemplo, predição de salário, preço, idade, etc. Já os problemas de classificação baseiam-se em prever a categoria de uma observação dada ao algoritmo. Por exemplo, dada uma amostra, dizer se ela é normal ou faz parte de um ataque.

Atualmente existem diversas categorias em que os modelos de aprendizado de máquina podem se enquadrar. Uma delas é o aprendizado supervisionado. Neste caso, o algoritmo depende de um conjunto de dados rotulados, ou seja, os dados utilizados para o treinamento devem conter as respostas desejadas. Durante o treinamento, o algoritmo recebe o conjunto de dados de entrada e para cada amostra ele faz uma predição, e na sequência, com base no rótulo, a verificação do quão próximo ela foi do esperado. Desta forma, o algoritmo consegue fazer os ajustes necessários para que as predições sejam cada

vez mais próximas aos rótulos.

No entanto, rotular os dados manualmente é uma tarefa demorada e custosa [30]. Para mitigar esse problema, existe a categoria dos algoritmos não supervisionados, como é o caso dos algoritmos *clustering*. Neste caso, o objetivo é extrair informações valiosas sobre o conjunto de dados, descobrindo padrões ocultos ou agrupamentos de dados sem a necessidade de intervenção humana. Por exemplo, dado um tráfego de rede com diversos tipos de ataque, um algoritmo não supervisionado pode ser utilizado para dividir esse conjunto em diferentes categorias. Para isso ele irá buscar padrões e semelhanças entre os ataques, podendo fazer a distinção entre eles sem a necessidade de um rótulo para validação [1].

Existe também a categoria dos algoritmos semi-supervisionados, como pode ser o caso dos algoritmos *One-Class*, que ao invés de classificar uma instância em um dos múltiplos rótulos pré-definidos, um único padrão é modelado e utilizado para distinguir se uma nova instância pertence ou não ao padrão aprendido. Esse modelo é útil para detectar anomalias em diferentes conjuntos de dados. Dispositivos IoT têm um comportamento padronizado e especializado, realizando tarefas com um uso bem definido, portanto a utilização de um algoritmo *One-Class* para detecção de anomalias pode ser viável nestes casos, sem a necessidade de um especialista para fazer a rotulação dos ataques manualmente [31].

2.3.1 Redes Neurais

As Redes Neurais (do inglês *Neural Networks*) são uma subárea do aprendizado de máquina, que simula a maneira que as sinapses funcionam no cérebro humano. Enquanto as abordagens tradicionais de computação utilizam de séries de blocos para executar as tarefas, as redes neurais utilizam redes compostas por nós (simulando os neurônios) e arestas (simulando as sinapses) para realizar o processamento dos dados [32].

O primeiro modelo de rede neural foi proposto por Warren McCulloch e Walter Pitts em 1943 [33], que descreveram como os neurônios devem funcionar. Além disso, modelaram suas ideias por meio de uma rede neural simples com circuitos elétricos. Esse modelo foi precursor para o desenvolvimento das pesquisas na área.

As pesquisas aceleraram rapidamente, até que, em 1975, Kunihiko Fukushima apresentou pela primeira vez o conceito de rede neural multicamadas [34]. O conceito das redes neurais multicamadas se mantém até os dias atuais e podem ser representadas conforme observado na Figura 1.

[1] <https://www.ibm.com/cloud/learn/machine-learning>

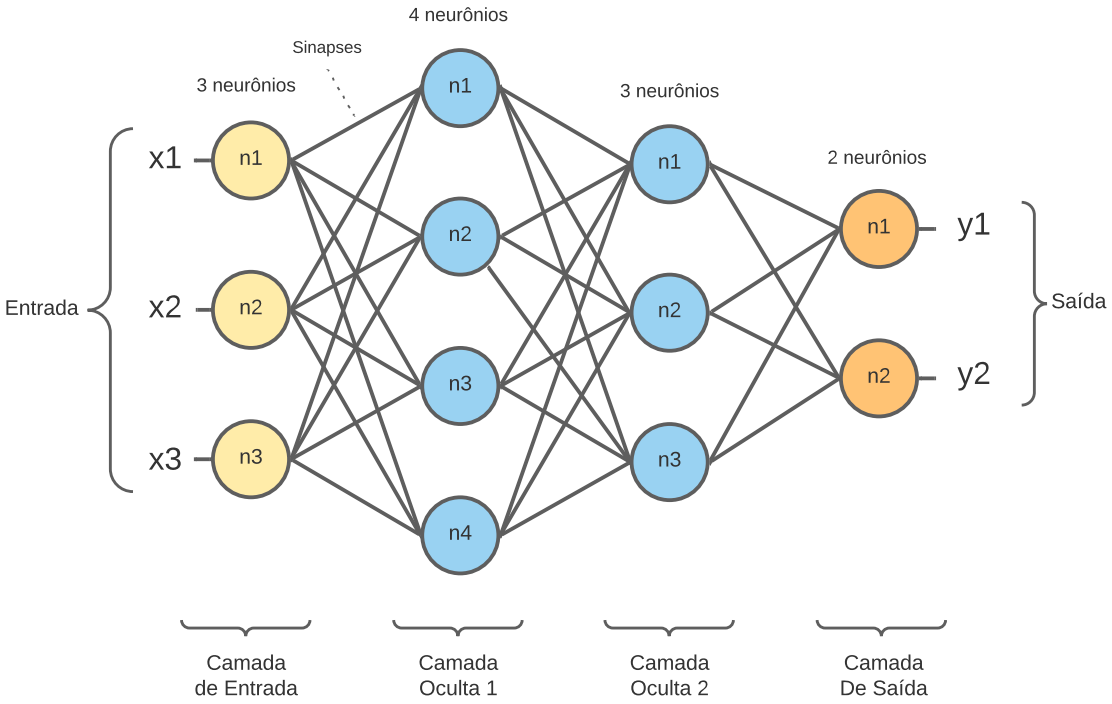

Figura 1 – Rede neural multicamadas

Usualmente as camadas são classificadas em três diferentes grupos:

· **Camada de Entrada:** Onde é apresentado o conjunto de dados de entrada para a rede.

· **Camadas Ocultas:** Onde é realizada a maior parte do processamento, através das conexões ponderadas por pesos.

· **Camada de Saída:** Onde é apresentado o resultado final do processamento.

As redes neurais multicamadas são projetadas de forma que os neurônios sejam organizados em duas ou mais camadas de processamento, visto que sempre existirá ao menos uma camada de entrada e uma de saída. Os neurônios se conectam por meio de arestas, e através destas conexões, os dados de entrada são processados pela rede até que se alcance a camada de saída.

2.3.2 Autoencoder

O Autoencoder é um tipo de Rede Neural utilizado, geralmente, para extrair as principais características de um conjunto de dados de maneira eficiente. Os primeiros conceitos de Autoencoder foram originalmente propostos por LeCun em 1987 [35]. Após

a publicação deste trabalho, outros pesquisadores da época publicaram artigos utilizando o Autoencoder para redução de dimensionalidade como forma de extração das principais características dos conjuntos de dados [36].

Com o passar dos anos, foram propostas diversas arquiteturas de Autoencoder, como: Autoencoders Padrão, Autoencoder Convolucional, Autoencoder Esparso, Autoencoder Denoising, Autoencoder Profundo, Autoencoders Variacionais, etc. Cada arquitetura proposta tem como objetivo solucionar problemas mais específicos com maior precisão e eficácia.

2.3.2.1 Aprendizado Profundo

O Autoencoder pode ser classificado como um tipo de Aprendizado Profundo (do inglês *Deep Learning*), que por sua vez é uma subárea derivada das redes neurais, que é caracterizada por possuir uma quantidade considerável de neurônios quando comparados com outros tipos de redes neurais [37]. Portanto, o aprendizado profundo permite que modelos compostos de diversas camadas de processamento resultem em uma maior capacidade de abstração e generalização dos dados [38].

Devido à maior capacidade de processamento e abstração dos dados, esses algoritmos visam resolver problemas mais complexos, ao passo que necessitam de treinamentos mais extensos para alcançar os resultados. Modelos de aprendizado profundo têm sido propostos por pesquisadores há mais de 50 anos. Entretanto, esses modelos se popularizaram recentemente com o avanço da tecnologia e da capacidade de processamento.

2.3.2.2 Autoencoder Profundo

Os Autoencoders Profundos têm como objetivo realizar a compressão dos dados de entrada, que na prática consiste no aumento da relevância das características que mais impactam na reconstrução das amostras, e posteriormente a reconstrução dos dados comprimidos na camada de saída, de forma que a reconstrução seja o mais similar possível aos dados de entrada. Para isso, ele é composto por duas redes simétricas, sendo a primeira metade responsável pela codificação e a segunda metade responsável pela decodificação dos dados [36, 39, 40].

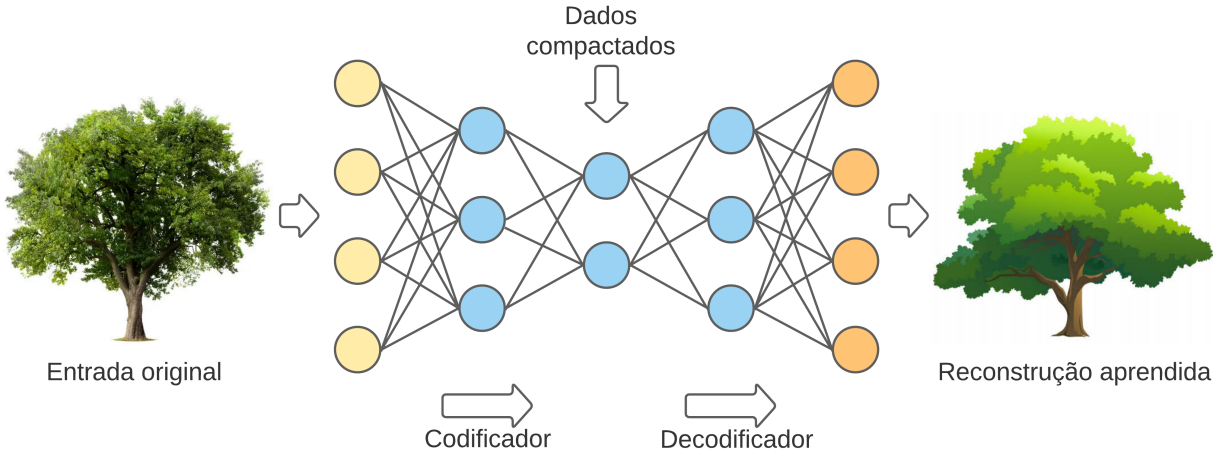

Figura 2 – Autoencoder Profundo

A Figura 2 representa a estrutura de um Autoencoder Profundo. Como é possível visualizar, o Codificador é responsável por realizar a redução de dimensionalidade, que consiste no processo de redução do número de atributos que descrevem alguns dados. É importante que esta etapa de compactação mantenha os principais atributos, de forma que o Decodificador consiga realizar a reconstrução a partir dos dados compactados [41].

Durante o treinamento, o principal objetivo é encontrar o melhor par codificador/decodificador, de forma que mantenha o máximo de informações ao codificar e consequentemente o mínimo de erro na reconstrução ao decodificar. Este processo pode ser resumido por:

$$(c^*, d^*) = arg\ min\ \epsilon(x, d(c(x))) \mid (c, d) \in C \times D \tag{2.1}$$

sendo C a família de codificadores e D a família de decodificadores, onde:

$$\epsilon(x, d(c(x))) \tag{2.2}$$

define a medida de erro entre os dados de entrada x e a reconstrução d(c(x)).

Sendo assim, o Autoencoder Profundo pode ser utilizado como uma ferramenta para detecção de anomalias. Para tal fim, o treinamento deve ser realizado apenas com os dados considerados normais. Posteriormente, o modelo será utilizado para reconstruir todos os tipos de dados. Os dados anormais, por sua vez, terão maior erro de reconstrução, sendo classificados como anômalos se o erro de reconstrução ultrapassar um limite fixo (*threshold*) pré determinado [2].

Portanto, para fazer a classificação, a função que calcula o erro baseado na comparação entre as amostras originais e reconstruídas, bem como o limiar fixo que determinará

[2] <https://www.tensorflow.org/tutorials/generative/autoencoder>

se a amostra é de fato anômala, devem ser definidos. Existem diversas funções que podem ser utilizadas para o cálculo do erro, nesse caso as mais utilizadas são o Erro Quadrático Médio (MSE), que foi a utilizada neste trabalho, e Erro Médio Absoluto (MAE). Já para o cálculo do limiar, normalmente são utilizadas técnicas de combinação entre média e desvio padrão do erro calculado [42].

2.4 Trabalhos Relacionados

Existem diversos métodos de detecção de anomalias com diferentes níveis de complexidade. Os Autoencoders, especialmente, têm sido amplamente utilizados, e seu funcionamento consiste basicamente no codificador mapear os dados originais em uma dimensão inferior e descobrir as características mais relevantes que representam o comportamento normal. A partir deste ponto, existem diversas técnicas que são utilizadas para classificar um comportamento como normal ou anômalo. A representação dos dados originais na dimensão inferior pode ser construída através do aprendizado supervisionado, aprendizado semi-supervisionado e aprendizado não supervisionado.

De maneira supervisionada, Vu et al. [43] utilizaram *Multi-distributed Variational Autoencoder (MVAE)* para representação dos dados normais e anômalos em duas diferentes áreas na camada central do modelo, conhecida também como camada de *bottleneck*. O principal objetivo foi deixar o tráfego mais distinguível entre as classes, e para isso adicionaram as informações dos rótulos das amostras na função de perda (*Kullback-Leibler*) do Variational Autoencoder (VAE). Essas informações dos rótulos permitem que os MVAEs dividam as amostras de dados de rede em diferentes classes com diferentes regiões na camada de *bottleneck*. Com as amostras de tráfego de rede mais distinguíveis, outros algoritmos de detecção de anomalias e intrusão foram utilizados para fazer a classificação. O modelo proposto foi avaliado em dois conjuntos de dados de segurança de rede e mostrou um desempenho promissor.

De maneira semi-supervisionada, Cao et al. [44] propuseram dois Autoencoders regularizados, denominados Shrink AE (SAE) e Dirac Delta VAE (DVAE), para capturar os comportamentos de dados normais. Os autores assumiram que apenas amostras normais estão disponíveis para treinamento, e nenhum dado anômalo pode ser usado para estimar hiperparâmetros. Esses Autoencoders regularizados visam solucionar o problema de identificação de anomalias em dados de rede de alta dimensão. SAE e DVAE têm como principal objetivo tornar os tráfego de redes mais distinguíveis. Foram então avaliados algoritmos de *one-class classification* (OCC), como: *Local Outlier Factor (LOF), Centroid (CEN), Kernel Density Estimation (KDE), Mean Distance (MDIS), One-class Support Vector Machine (OCSVM)* utilizando tanto os dados produzidos pelo SAE e DVAE, quanto os dados brutos. A comparação mostrou que de fato os resultados foram melhores com a utilização do modelo.

Já Nguyen et al. [45] propuseram um *Clustering-based deep AutoEncoder* (CAE) para representação dos dados de entrada no espaço latente e identificação de anomalias de forma semi-supervisionada. A proposta consiste em uma combinação de uma variação do algoritmo *k-means clustering* e um Autoencoder. Eles se basearam na suposição de que os dados normais podem possuir diversas subclasses, apesar de compartilharem características que os tornam normais como um todo. O funcionamento do modelo consiste no treinamento apenas utilizando dados normais. Inicialmente, o Autoencoder é responsável pela representação dos dados no espaço latente através do decodificador, e posteriormente os dados são divididos em diferentes *clusters* de acordo com cada subclasse. Durante o processo de classificação, os dados de entrada são representados no espaço latente, e posteriormente é calculada a distância entre estes dados e o *cluster* mais próximo. Se a distância entre eles for maior que um determinado *threshold*, então ela é classificada como anômala.

Por fim, Meidan et al. [21] utilizaram *Deep Autoencoders* para detecção de anomalias por meio de aprendizado semi-supervisionado. A proposta se baseou na detecção de ataques em *botnets* IoT, focando principalmente na fase em que o botmaster já enviou o comando de ataque aos bots e eles estão executando esta instrução. Para tanto, foi proposta a utilização de um Autoencoder por dispositivo. Foram extraídas 115 características para realização do treinamento em 5 janelas de tempo diferentes. O Autoencoder então foi treinado para reconstrução destas 115 características utilizando somente os dados normais. Durante o processo de classificação, caso a diferença entre os dados de entrada e reconstrução deles seja maior que um determinado *threshold*, então esses dados são considerados anômalos.

Este trabalho propõe a utilização do aprendizado semi-supervisionado através de *Deep Autoencoders*. A proposta tem como foco a detecção de eventos que são menos nítidos no tráfego de rede, como *file download* e mensagens de *C&C*. Para isso, para cada dispositivo conectado são extraídas 48 características do tráfego de rede em janelas de 60 segundos. O principal objetivo é verificar se estas características escolhidas são capazes de fazer o modelo identificar anomalias mesmo em casos sem grande impacto no tráfego de rede. Diferente do proposto no trabalho de Meidan et al. [21], o objetivo não se limita em detectar a última etapa do ataque realizado por uma *botnet*. Para isso, selecionamos diferentes características em janelas de tempo diferentes.

3 PROCEDIMENTOS E MÉTODOS

3.1 Solução Proposta

O método proposto neste trabalho consiste na detecção de ataques em redes IoT utilizando métodos baseados na análise do comportamento do tráfego de rede dos dispositivos.

Com esse propósito, é utilizado um Autoencoder Profundo para cada dispositivo conectado na rede. Estes modelos são previamente treinados utilizando somente dados normais, e a ideia é manter os dispoisitivos conectados na rede, para que o algoritmo possa ser retreinado de modo que se adapte aos comportamentos normais do dispositivo. Portanto, o método de funcionamento consiste basicamente nas seguintes etapas: 1) coleta dos pacotes, 2) pré-processamento e extração de características, 3) treinamento, 4) classificação. Uma visão mais completa pode ser observada na Figura 3.

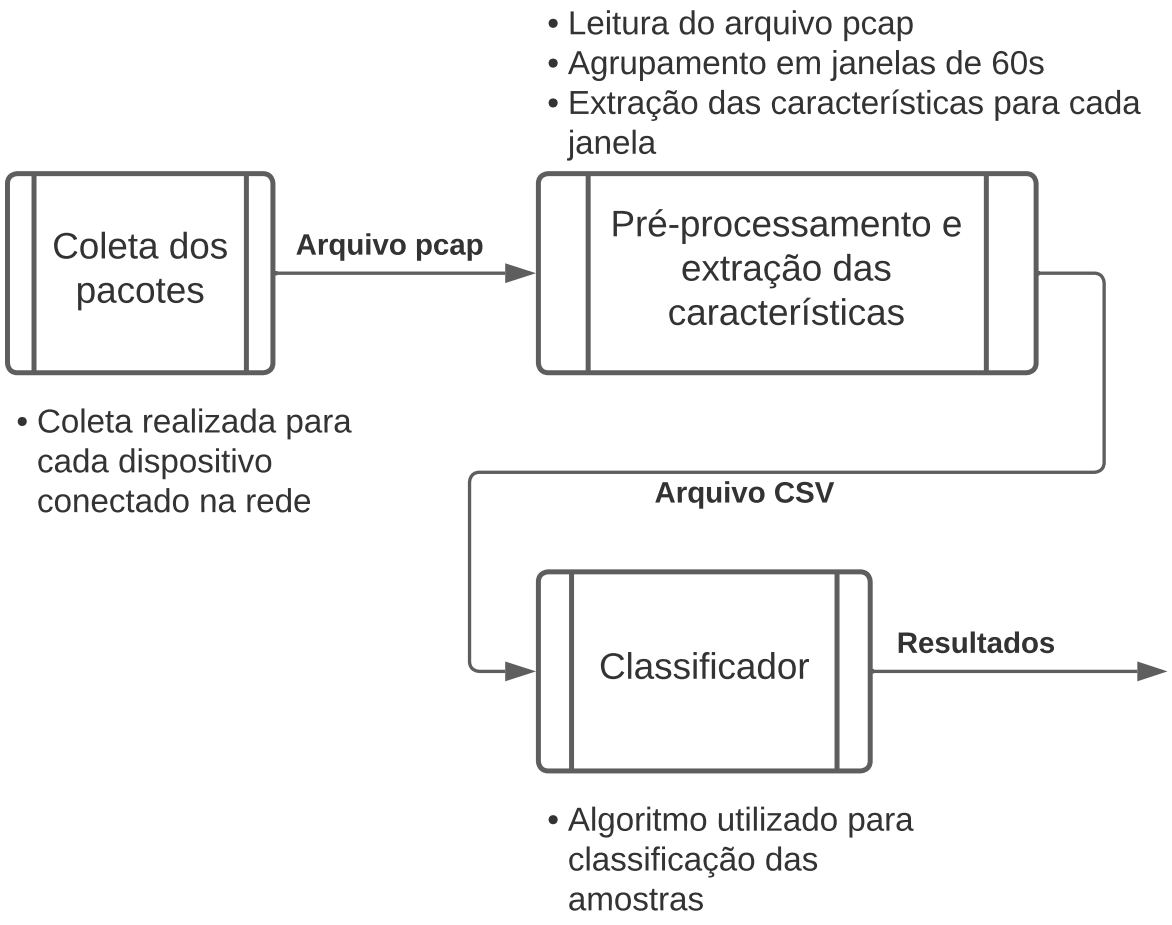

Figura 3 – Diagrama com visão geral da proposta

Inicialmente, são capturados os pacotes do tráfego na rede (no formato *pcap*), e para garantir que os dados de treinamento são livres de amostras maliciosas, os dados devem ser coletados imediatamente após a instalação do dispositivo na rede.

Após a coleta, os pacotes são organizados em janelas de 60 segundos da seguinte forma: cada pacote é agrupado com todos os próximos pacotes contidos dentro da uma janela de 60 segundos conforme os tempos de captura. Um exemplo de agrupamento pode ser observado na Figura 4, sendo cada pacote representado por um quadrado. Como pode ser observado, um conjunto de n pacotes geram n janelas de tempo.

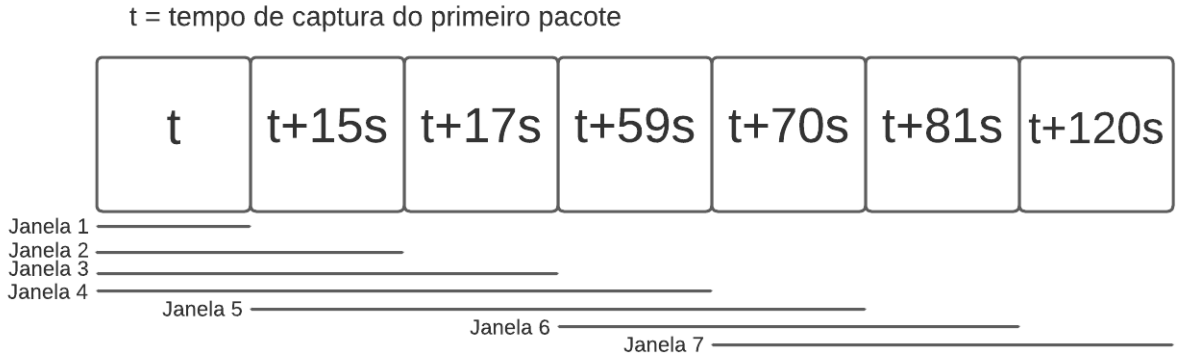

Figura 4 – Exemplo de agrupamento dos pacotes em janelas de 60 segundos

Posteriormente é realizada a extração das características que irão compor as amostras apresentadas ao modelo. Cada amostra é constituída por 48 características extraídas de cada janela capturada.

Por fim, as características são apresentadas aos algoritmos para classificação das amostras. O conjunto de dados é separado em 3 partes, sendo a primeira para treinamento, a segunda para otimização do *threshold* e do número de épocas, que consiste no número de vezes que o Autoencoder será treinado, e a última para teste.

Para a realizar a classificação das amostras, o *threshold* foi determinado pela soma da média com o desvio padrão do erro quadrático médio do conjunto de dados de otimização, representado pela equação 3.1.

$$tr = \overline{MSE}_{DSotm} + std(MSE_{DSotm}) \tag{3.1}$$

3.2 Características Extraídas

O principal objetivo na extração das características é encontrar dados que de fato se diferem em situações nas quais estão ocorrendo uma anomalia das situações com comportamento normal. Para isso, foram escolhidas características com o intuito de satisfazer

esse objetivo, mesmo para eventos menos intensos no tráfego de rede, como *File Download* e mensagens de *C&C*.

As características são extraídas partir de arquivos *pcap (Packet Capture)*, que contém os pacotes brutos da rede, pré-processados e agrupados em janelas de 60 segundos. Foram extraídas 48 características, sendo elas:

1. Estatísticas do tamanho dos pacotes, que normalmente sofrem alterações em momentos de ataque.
2. Quantidade de pacotes, que em casos como *DDoS* sofrem um grande aumento no volume.
3. Estatísticas do *TTL (Time To Live)* dos pacotes, que representa o número máximo de saltos entre máquinas que o pacote pode realizar na rede.
4. Entropia das portas de origem e destino, que sofrem um grande aumento ou grande queda em casos como *PortScan* e *DDoS*.
5. Contagem da ocorrência de *flags* TCP, que podem sofrer alterações em ataques com pacotes TCP.

Os dados, quantidade e agrupamento das características extraídas podem ser visualizados de forma geral na Tabela 1.

Tabela 1 – Características extraídas

Dados	**Estatísticas**	**Agregado por**	**Numero total de características**
Tamanho do pacote	Mínimo, máximo, média, variância, mínima/máximo, média/máximo	Pacotes de entrada e saída	12
Quantidade de pacotes	Número	Pacotes de entrada e saída	2
Time to live (TTL)	Mínimo, máximo, média, variância, mínima/máximo, média/máximo	Pacotes de entrada e saída	12
Portas de destino	Entropia	Pacotes de entrada e saída	2
Portas de origem	Entropia	Pacotes de entrada e saída	2
Flags TCP (ACK, SYN, FIN, PSH, RST)	Quantidade de ocorrências	Pacotes de entrada e saída	10
Códigos *ICMP (None, 0, 1, 3)*	Quantidade de ocorrências	Pacotes de entrada e saída	8

4 EXPERIMENTOS E RESULTADOS

4.1 Experimentos

Nos experimentos realizados neste trabalho, foram utilizados os conjuntos de dados de Anthi et al. [46] e IoT-23 [47]. Foram comparados três algoritmos durante o processo de classificação, sendo eles, o Autoencoder, o *One-Class SVM (Support Vector Machine)* com *kernel RBF* e o *LOF (Local Outlier Factor)* com valor de 20 vizinhos. Os parâmetros do *SVM* e do *LOF* não foram otimizados durante o processo de treinamento.

Para cada cenário, foram utilizados 3 conjuntos de dados separados. No treinamento e otimização, foram utilizados dois diferentes conjunto de dados, ambos contendo somente dados normais. O primeiro é o *dataset* de treinamento (*DStre*), responsável somente pela realização do treinamento dos modelos. O segundo é o *dataset* de otimização (*DSotm*), que por sua vez foi utilizado, no Autoencoder, para otimização do *threshold* e do número de épocas de treinamento. Por fim, o *dataset* de teste (*DStst*) é um conjunto de dados normais e anômalos, utilizado para fazer a avaliação do modelo de acordo com as métricas.

Nos experimentos, a arquitetura do Autoencoder foi constituída por 4 camadas ocultas no codificador, sendo elas definidas em tamanhos decrescentes de 75%, 50%, 33% e 25% do tamanho da camada de entrada. O decodificador por sua vez foi constituído por um espelhamento do codificador, iniciando em 33%. Na construção do algoritmo, foram utilizadas as bibliotecas Tensorflow e Keras, para otimização do número de épocas foi utilizado o *callback* EarlyStopping com monitoramento na *loss* e patience 5. Na Tabela 2 é possível visualizar todos os parâmetros utilizados na construção do modelo.

Tabela 2 – Parâmetros utilizados na construção do Autoencoder

Normalizador	z-score
Função de ativação	tangente hiperbólica
Função de Loss	erro quadrático médio
Otimizador	SGD
Número máximo de épocas	2000
Learning rate	0,01
Batch size	32

4.2 Métricas Utilizadas para Avaliação de Desempenho

Para medir a performance dos classificadores, foram aplicadas as seguintes métricas: precisão (*Precision*), revocação (*Recall*), *F1* e taxa de falsos positivos (*False Positive Rate - FPR).*

O significado das siglas apresentadas nas equações são: TP representando os verdadeiro positivos (*True Positives*), ou seja, as amostras corretamente classificadas como anômalas. TN representa os verdadeiro negativos (*True Negatives*), que são as amostras acertadamente classificadas como normais. Já o FP são os falso positivos (*False Positives*), que são as amostras nas quais o algoritmo classificou como anômalas, mas não são. Os FN, por sua vez, são os falso negativos (*False Negatives*), representados pelas amostras marcadas por serem normais, no entanto são anômalas.

Cada uma das métricas são definidas da seguinte forma:

1. ***Precision***: representa a porcentagem de amostras verdadeiramente classificadas como anômalas

$$precision = \frac{TP}{TP + FP} \tag{4.1}$$

2. ***Recall***: representa a efetividade em identificar amostras anômalas

$$recall = \frac{TP}{TP + FN} \tag{4.2}$$

3. ***F1***: representa a média harmônica entre a precisão e o *recall*

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{4.3}$$

4. ***FPR***: representa a taxa de falsos positivos, ou seja, quantos foram classificados como anômalos, mas eram normais

$$FPR = \frac{FP}{FP + TN} \tag{4.4}$$

4.3 Conjunto de dados de Anthi et al.

No trabalho de Anthi et al. [46] os *datasets* foram montados em laboratório com o objetivo de simular ataques reais a dispositivos IoT. Posteriormente foram realizadas classificações para determinar os tipos dos ataques. Os conjuntos de dados disponibilizados foram divididos em diferentes arquivos, o primeiro contendo somente dados normais e o segundo uma mescla de dados normais e ataques.

Como parte das etapas de processamento deste trabalho, inicialmente não foi necessário fazer a extração dos dados dos arquivos pcap, uma vez que eles já estavam extraídos e disponibilizados em arquivos no formato ARFF. Portanto, foi realizado somente o pré-processamento para junção dos dados normais e anômalos em um único conjunto de dados, ordenação por tempo de captura e posteriormente o agrupamento por janelas de 60 segundos. Por fim, foram extraídas as 48 características estabelecidas para cada janelas e disponibilizadas para o treinamento dos algoritmos.

Do conjunto de dados extraído, foi possível formar 4 *datasets* sendo um para cada dispositivo: Hive, Lifx, SmartThings e TPLinkCam. Os ataques contidos nos conjuntos de dados variam entre *Address Resolution Protocol (ARP) spoofing, Media Access Control (MAC) address spoofing, Man-In-The-Middle (MITM), port scanning, capturing handshakes, sniffing* e *Denial of Service (DoS).*

4.4 Conjunto de dados IoT-23

IoT-23 [47] são conjuntos de dados de tráfego de rede de dispositivos da Internet das Coisas, que contém capturas com diferentes *malwares* e 3 capturas de tráfego normal dos dispositivos Amazon Echo, Philips Hue e Somfy door lock. Esses conjuntos de dados foram criados pelo laboratório Avast AIC com financiamento da Avast Software. O objetivo principal da criação foi oferecer grandes conjuntos de dados de tráfegos de rede reais de dispositivos IoT, para que pesquisadores pudessem desenvolver algoritmos de aprendizado de máquina para detecção de *malwares*.

Durante a criação dos conjuntos de dados, para cada cenário malicioso, foi executado um *malware* específico em um Raspberry Pi utilizando vários protocolos e diferentes ações. Já os tráfegos de rede capturados dos cenários normais foram obtidos a partir de dispositivos IoT reais, o que permitiu capturar e analisar o comportamento real destes dispositivos na rede. Ambos os casos foram executados em ambiente de rede controlado com conexão de Internet irrestrita, com o objetivo de simular um cenário real como qualquer outro dispositivo IoT.

Portanto, os conjuntos de dados disponibilizados são basicamente de dois tipos: o primeiro contém os tráfegos maliciosos e o segundo somente os normais. No caso dos conjuntos de dados contendo tráfegos maliciosos, a equipe do laboratório executou um analisador de rede chamado Zeek sobre o arquivo pcap original. Esse analisador é responsável por verificar os fluxos de rede e rotular os fluxos maliciosos. No fim do processo de rotulação, foi gerado um arquivo chamado "conn.log.labeled", contendo os tempos de captura para cada fluxo malicioso.

Com isso, para cada cenário malicioso, foram disponibilizados dois arquivos: o pcap original do tráfego capturado e o "conn.log.labeled", contendo os tempos de captura para cada fluxo malicioso.

Com os conjuntos de dados normais, e maliciosos contendo os tempos de captura para cada fluxo malicioso, foi possível, neste trabalho, fazer a captura dos pacotes maliciosos e posteriormente a injeção desses pacotes nos fluxos de rede normais. Para assim fazer a avaliação dos algoritmos para cada cenário.

Assim sendo, para que pudéssemos capturar todos os pacotes que faziam parte do tráfego malicioso, foi necessário percorrer cada fluxo de tempo contido no arquivo "conn.log.labeled"pelo arquivo pcap e capturar todos os pacotes contidos na janela, desde que os IPs e portas fossem do mesmo fluxo. Um exemplo de fluxo selecionado pode ser visualizado na Figura 5.

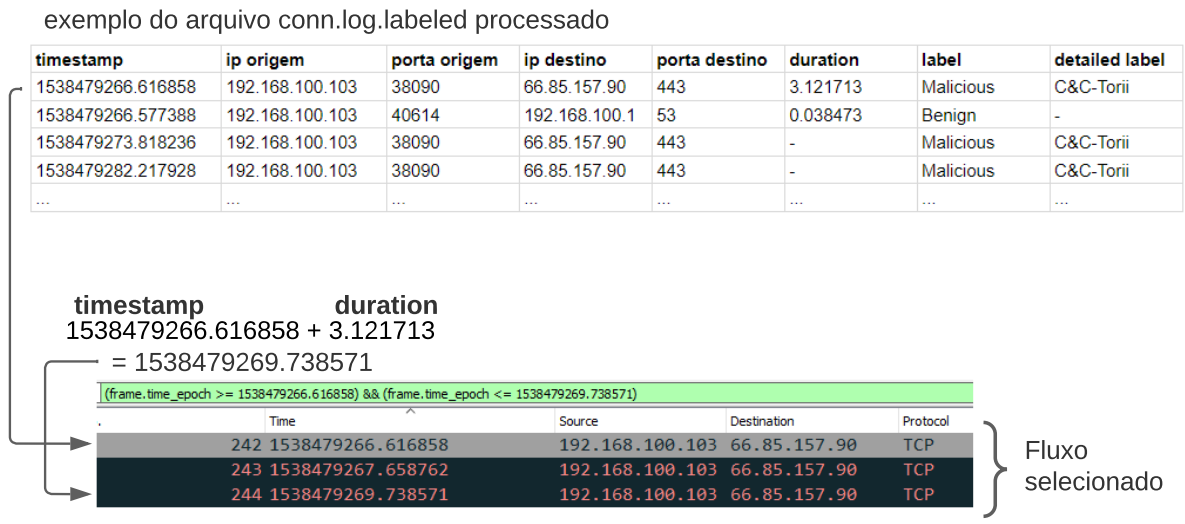

Figura 5 – Rotulação dos pacotes maliciosos

Após a rotulação de todos os pacotes maliciosos, foi realizada a injeção do ataque capturado no meio do fluxo de rede normal dos dispositivos IoT. Para tal fim, foi mantido o espaçamento dos tempos de coleta dos pacotes maliciosos, mas o tempo de captura do primeiro pacote do fluxo foi definido de acordo com os tempos de coleta no tráfego normal.

Por fim, foi realizado o agrupamento dos pacotes em janelas de 60 segundos e rotulada cada janela como anômala caso contivesse ao menos um pacote malicioso. Um exemplo de injeção do conjunto de dados CTU-IoT-Malware-Capture-7-1 no dispositivo Philips HUE pode ser observado na Figura 6.

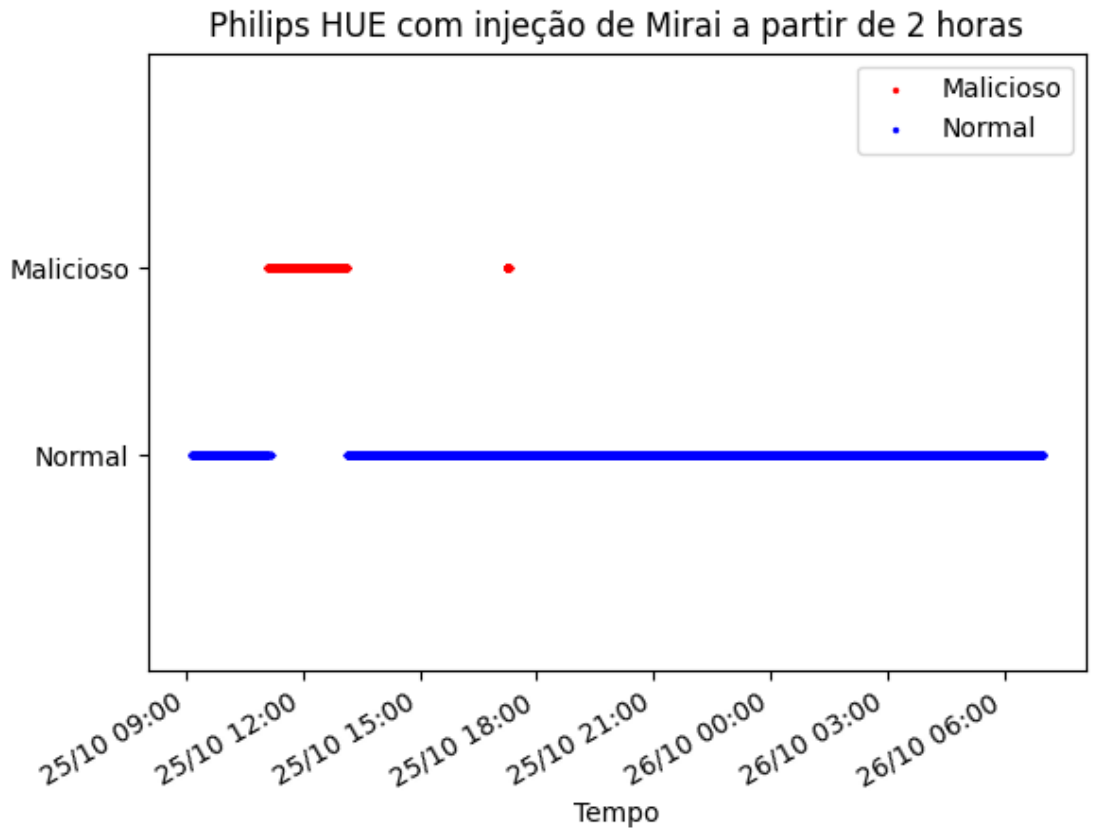

Figura 6 – Injeção do ataque no tráfego do dispositivo IoT

Foram realizadas 4 injeções de ataque no total, todas elas no dispositivo Philips HUE já que o monitoramento do dispositivo durou 24 horas, permitindo a inserção de ataques sem que eles dominassem parte muito significativa do período analisado. Os ataques injetados foram Linux.Mirai (C&C-HeartBeat, DDoS, Okiru), Torii (C&C-Torii), Mirai (C&C, DDoS, PartOfAHorizontalPortScan) e Trojan (C&C-FileDownload, FileDownload). Representados, respectivamente, pelos conjuntos de dados CTU-IoT-Malware-Capture-7-1, CTU-IoT-Malware-Capture-20-1, CTU-IoT-Malware-Capture-34-1 e CTU-IoT-Malware-Capture-42-1.

4.5 Resultados

Após a coleta e processamento de todos os conjuntos de dados, foram realizadas comparações entre os diferentes algoritmos para cada conjunto de dados. Na Figura 7, podemos observar o *boxplot* de cada algoritmo para o F1 de todos os *datasets*. Como é possível visualizar, os resultados do Autoencoder apresentam os valores mais altos com menor dispersão. Além disso o *outlier* se apresenta muito próximo do resto dos valores. Diferente do que acontece no *SVM* por exemplo, o qual apresenta alta dispersão dos resultados e um *outlier* significante.

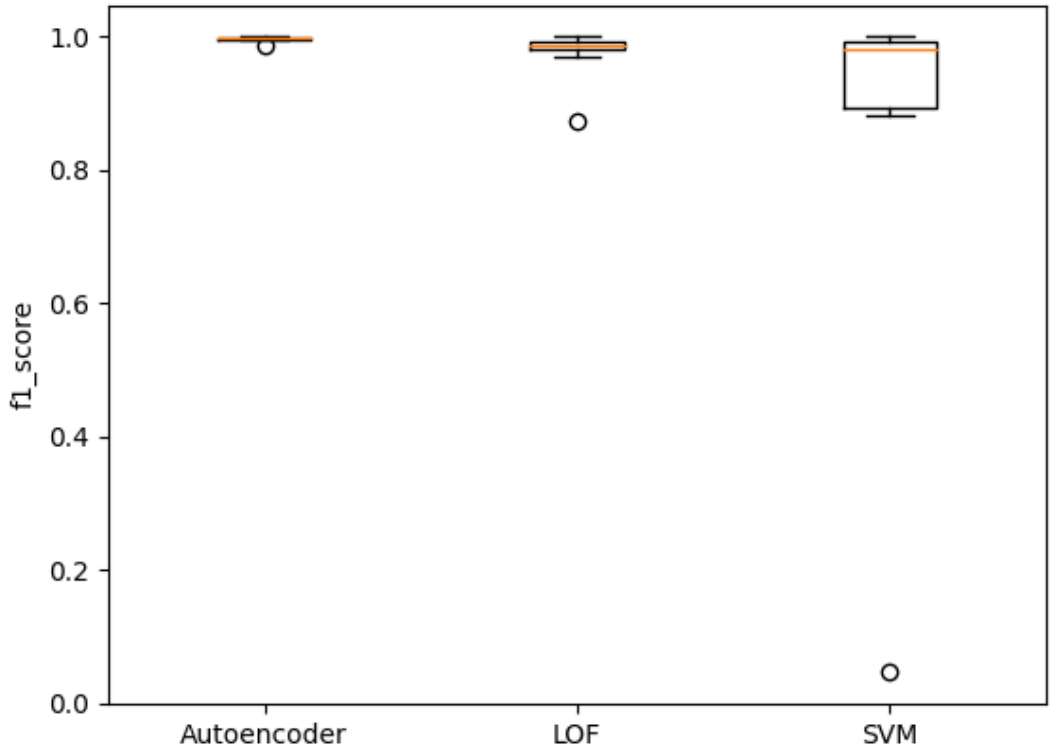

Figura 7 – *Boxplots* de cada classificador para todos os conjuntos de dados

Na Tabela 3, são apresentadas as métricas para cada classificador em cada um dos conjuntos de dados utilizados durante o experimento. Os melhores resultados para cada cenário estão destacados em negrito.

Tabela 3 – Comparação de métricas para cada cenário

Dataset	Algoritmo	Precision	Recall	FPR
CTU-IoT-Malware-Capture-7-1	**Autoencoder**	**0,9999**	0,9999	**0,0130**
	LOF	0,9997	**1,0000**	0,0955
	SVM	0,9998	0,9999	0,0260
CTU-IoT-Malware-Capture-20-1	**Autoencoder**	**0,9997**	0,9945	**0,0014**
	LOF	0,9833	**1,0000**	0,1093
	SVM	0,8925	0,0241	0,0188
CTU-IoT-Malware-Capture-34-1	Autoencoder	0,9978	0,9997	0,0664
	LOF	0,9967	**1,0000**	0,1029
	SVM	**0,9991**	0,9772	**0,0267**
CTU-IoT-Malware-Capture-42-1	**Autoencoder**	**0,9984**	0,9750	**0,0005**
	LOF	0,7760	**1,0000**	0,1095
	SVM	0,8278	0,9750	0,0769
Hive	**Autoencoder**	**0,9968**	1,0000	**0,0049**
	LOF	0,7760	1,0000	0,1095
	SVM	0,8278	0,9750	0,0769
Lifx	**Autoencoder**	**0,9984**	0,9750	**0,0005**
	LOF	0,9411	1,0000	0,0985
	SVM	0,9828	1,0000	0,0275
SmartThings	**Autoencoder**	**0,9919**	1,0000	**0,0222**
	LOF	0,9703	1,0000	0,0832
	SVM	0,9498	1,0000	0,1439
TPLinkCam	**Autoencoder**	**0,9958**	1,0000	**0,0126**
	LOF	0,9738	1,0000	0,0798
	SVM	0,9911	1,0000	0,0264

Os resultados apresentados indicam que o Autoencoder teve bons resultados em todas as métricas selecionadas, apresentando as melhores precisões se comparado com os outros algoritmos, todas elas acima de 99%. Enquanto isso, o LOF teve os melhores resultados para o *recall* na maioria dos casos. Além disso, o Autoencoder se mostrou com baixa quantidade de falsos positivos, demonstrando uma alta eficácia em identificar as amostras anômalas.

No caso do Autoencoder, foi realizado o cálculo do *threshold*, utilizado para realizar a classificação das amostras. Os valores dos *thresholds* podem ser visualizados de forma geral nas Figuras 8 e 9, representando respectivamente os conjuntos de dados de Anthi e IoT-23.

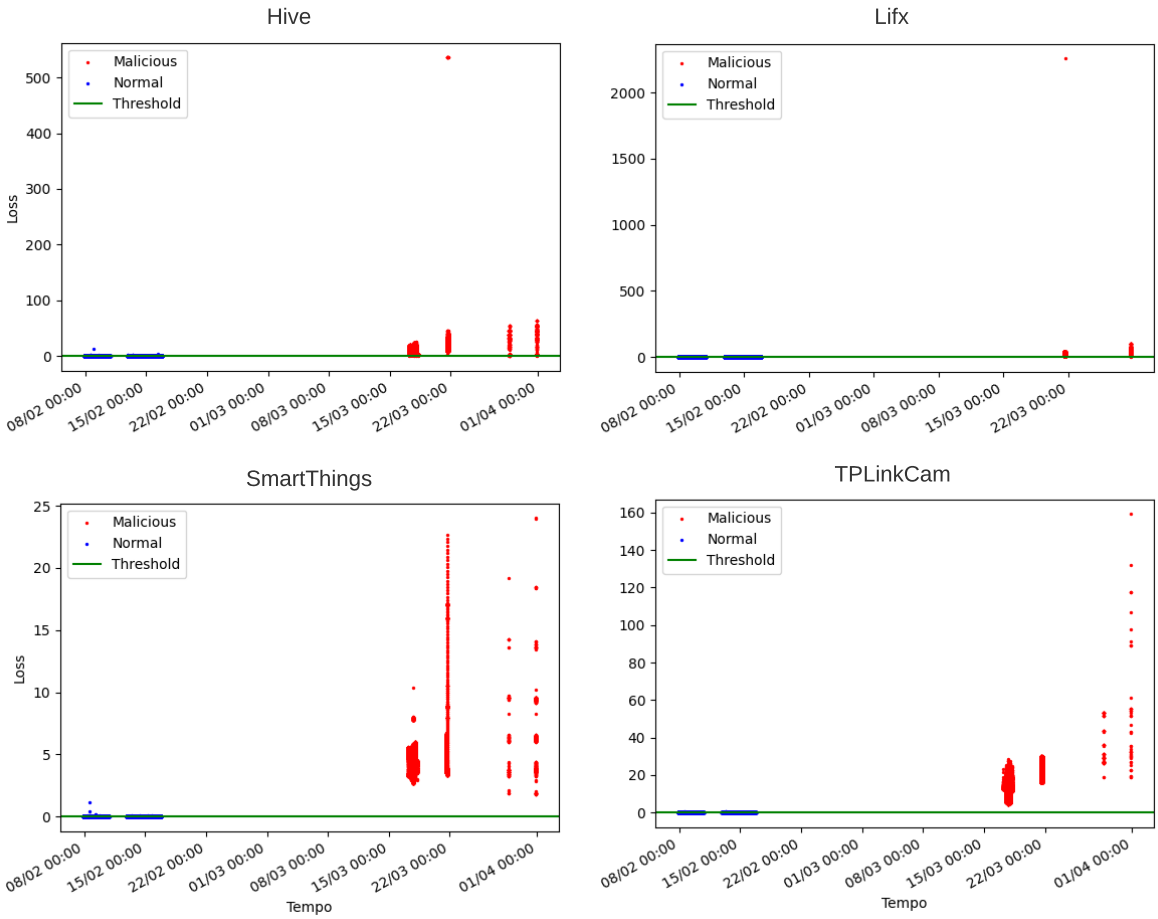

Figura 8 – Gráficos da perda ao longo do tempo - Anthi et al.

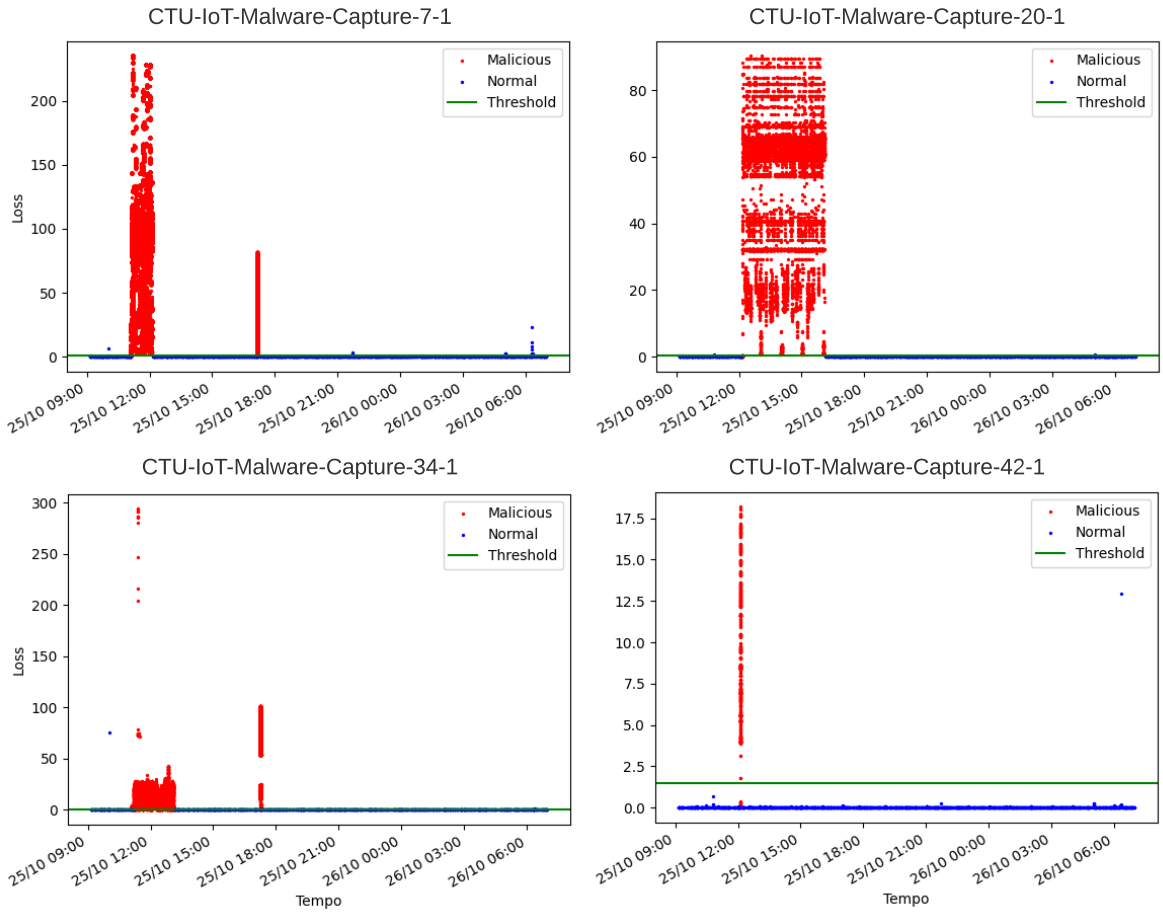

Figura 9 – Gráficos da perda ao longo do tempo - IoT-23

O eixo y é representa a *Loss*, que consiste no erro entre os valores reconstruídos pelo Autoencoder e os valores originais da amostra. A função de erro utilizada no experimento foi o erro quadrático médio. Portanto, ao chegar uma nova amostra, ela é reconstruída pelo Autoencoder e posteriormente é feito o cálculo do erro quadrático médio entre os valores originais da amostra e os valores reconstruídos pelo Autoencoder. O resultado desse cálculo é a *Loss*, sendo assim, quanto maior a *Loss*, maior a probabilidade da amostra ser maliciosa.

Nas figuras é possível observar as amostras maliciosas representadas pela cor vermelha, as normais pela cor azul e o *threshold* pela cor verde. Todas as amostras com a *Loss* maior que o valor do *threshold* são classificadas como maliciosas pelo modelo.

Portanto, como é possível visualizar, os resultados se mostraram satisfatórios. Em todos os casos, o *threshold* fica o mais próximo possível de separar as amostras normais das maliciosas. Vale ressaltar o caso do conjunto de dados CTU-IoT-Malware-Capture-42-1, no qual foi possível identificar um ataque de *FileDownload*, normalmente dificilmente detectável devido a baixa variação nos valores encontrados no tráfego de rede.

Na Figura 10 é possível ver um exemplo extraído do conjunto de dados Lifx, contendo a comparação das características extraídas entre uma amostra normal e uma anômala. Os valores estão normalizados utilizando Z-Score e é possível ver uma diferença de valores na maioria das características, o que leva os algoritmos de aprendizado de máquina a identificarem as diferenças entre as amostras. Grande parte dos casos apresentaram gráficos parecidos, o que indica que as características selecionadas pareceram adequadas para a classificação das amostras.

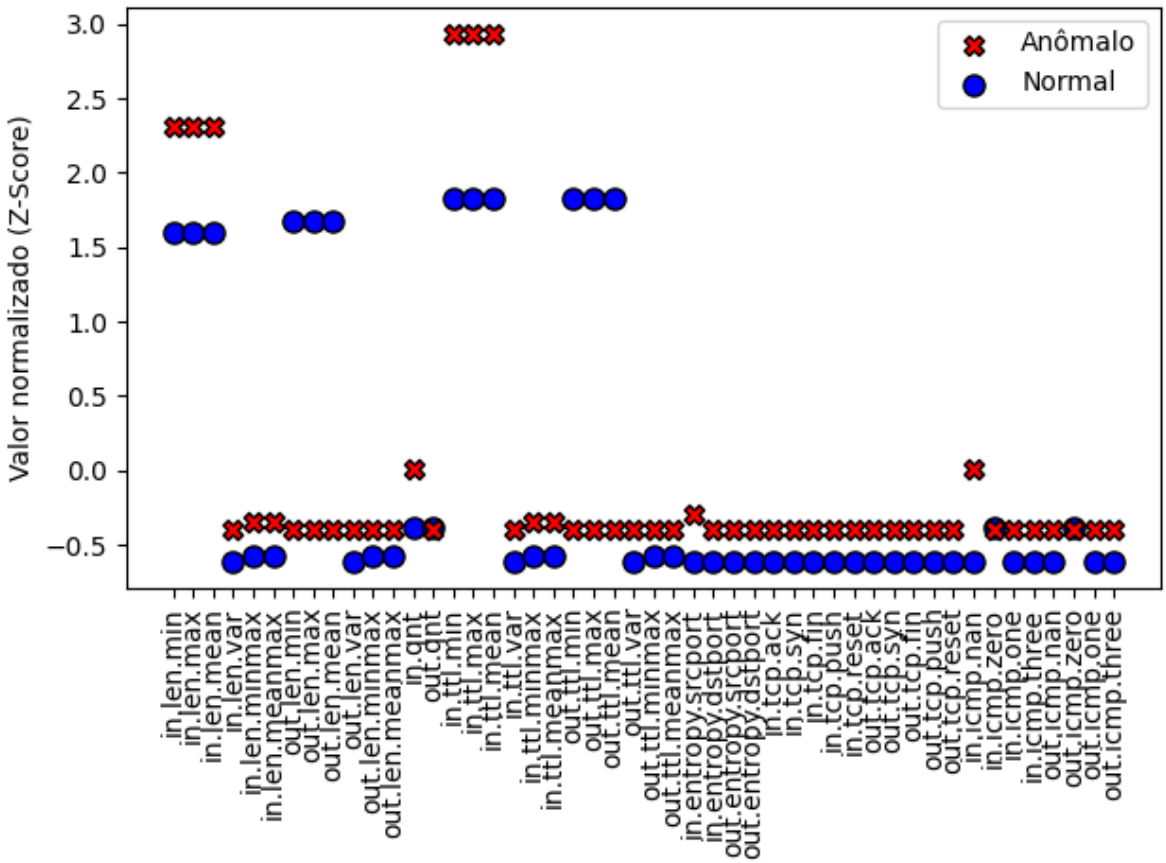

Figura 10 – Comparação das características extraídas

5 CONCLUSÃO

O paradigma da Internet das Coisas tem sido amplamente utilizado na sociedade atual, tendo como grande desafio a confiabilidade, segurança e privacidade desses dispositivos. Portanto, são necessárias medidas para evitar que agentes mal intencionados prejudiquem outras pessoas e organizações por meio desses dispositivos.

Pesquisas envolvendo a detecção de *malwares* em ambientes *desktop* são comuns e facilmente encontradas na literatura. No entanto, com o crescimento de dispositivos IoT conectados e as particularidades que eles possuem, são necessárias adaptações para monitoramento desses dispositivos.

O sistema de deteção de anomalias baseado em rede proposto neste trabalho visa realizar as adaptações necessárias para que seja possível detectar os possíveis ataques a esses dispositivos com maior precisão. Para isso, o sistema proposto aborda desde a coleta dos pacotes de tráfego na rede, passando pela extração das principais características até a classificação dos modelos. Foram realizadas comparações entre três algoritmos de Aprendizado de Máquina, sendo eles o Autoencoder, *SVM* e *LOF*.

Durante os experimentos, foi possível observar que as características escolhidas certamente são viáveis para a detecção dos ataques, uma vez que na maioria dos casos, os três algoritmos comparados obtiveram um bom desempenho. Apesar disso, o Autoencoder se sobressaiu na maioria dos casos, apresentando uma precisão sempre maior que 99% e uma taxa de falsos positivos próxima de 1%.

Com relação ao Autoencoder, a equação escolhida para o cálculo de *threshold* pareceu adequada para solução do problema, uma vez que na maioria dos casos o valor do *threshold* separou o maior número possível de amostras normais das anômalas.

Por fim, foi possível observar que a solução proposta neste trabalho é capaz de identificar diferentes tipos de ataque em redes IoT, obtendo sucesso desde ataques que se manifestam de maneira mais agressiva, gerando um grande ruído nos dados extraídos do tráfego de rede, como *DDoS*, até os que geram menos alterações nos valores coletados, como *C&C* e *File Download*. Além disso foi possível constatar que é possível o modelo se adaptar para cada dispositivo conectado na rede, obtendo uma maior precisão na detecção dos ataques, uma vez que os padrões comportamentais do dispositivo podem ser aprendidos pelo modelo através do treinamento.

Para trabalhos futuros, desejamos avaliar a diminuição da janela de tempo de 60 segundos utilizada neste trabalho, podendo identificar os ataques com mais rapidez, além de realizar uma avaliação mais profunda da adaptação deste sistema em um cenário real, realizando a captura, processamento, classificação e retreino para cada dispositivo

conectado em tempo real.

REFERÊNCIAS

[1] BEZERRA, V. H. et al. Providing iot host-based datasets for intrusion detection research. In: SBC. *Anais do XVIII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*. [S.l.], 2018. p. 15–28.

[2] BEZERRA, V. H. et al. Iotds: A one-class classification approach to detect botnets in internet of things devices. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 19, n. 14, p. 3188, 2019.

[3] KOLIAS, C. et al. Ddos in the iot: Mirai and other botnets. *Computer*, IEEE, v. 50, n. 7, p. 80–84, 2017.

[4] BERTINO, E.; ISLAM, N. Botnets and internet of things security. *Computer*, IEEE, v. 50, n. 2, p. 76–79, 2017.

[5] ELRAWY, M. F.; AWAD, A. I.; HAMED, H. F. Intrusion detection systems for iot-based smart environments: a survey. *Journal of Cloud Computing*, SpringerOpen, v. 7, n. 1, p. 1–20, 2018.

[6] ZARPELÃO, B. B. et al. A survey of intrusion detection in internet of things. *Journal of Network and Computer Applications*, Elsevier, v. 84, p. 25–37, 2017.

[7] IOULIANOU, P. et al. A signature-based intrusion detection system for the internet of things. *Information and Communication Technology Form*, York, 2018.

[8] OTOUM, Y.; LIU, D.; NAYAK, A. Dl-ids: a deep learning–based intrusion detection framework for securing iot. *Transactions on Emerging Telecommunications Technologies*, Wiley Online Library, p. e3803, 2019.

[9] OZAY, M. et al. Machine learning methods for attack detection in the smart grid. *IEEE transactions on neural networks and learning systems*, IEEE, v. 27, n. 8, p. 1773–1786, 2015.

[10] PERERA, P.; OZA, P.; PATEL, V. M. One-class classification: A survey. *arXiv preprint arXiv:2101.03064*, 2021.

[11] MEIDAN, Y. et al. N-baiot—network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, IEEE, v. 17, n. 3, p. 12–22, 2018.

[12] GOKHALE, P.; BHAT, O.; BHAT, S. Introduction to iot. *International Advanced Research Journal in Science, Engineering and Technology*, v. 5, n. 1, p. 41–44, 2018.

[13] ABDUL-GHANI, H. A.; KONSTANTAS, D.; MAHYOUB, M. A comprehensive iot attacks survey based on a building-blocked reference model. *International Journal of Advanced Computer Science and Applications*, The Science and Information Organization, v. 9, n. 3, 2018. Disponível em: <http://dx.doi.org/10.14569/IJACSA.2018.090349>.

[14] COLITTI, W. et al. Evaluation of constrained application protocol for wireless sensor networks. In: IEEE. *2011 18th IEEE Workshop on Local & Metropolitan Area Networks (LANMAN)*. [S.l.], 2011. p. 1–6.

[15] SHERBURNE, M.; MARCHANY, R.; TRONT, J. Implementing moving target ipv6 defense to secure 6lowpan in the internet of things and smart grid. In: *Proceedings of the 9th Annual Cyber and Information Security Research Conference*. [S.l.: s.n.], 2014. p. 37–40.

[16] ABDUL-GHANI, H. A.; KONSTANTAS, D.; MAHYOUB, M. A comprehensive iot attacks survey based on a building-blocked reference model. *International Journal of Advanced Computer Science and Applications*, v. 9, n. 3, p. 355–373, 2018.

[17] DEOGIRIKAR, J.; VIDHATE, A. Security attacks in iot: A survey. In: *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. [S.l.: s.n.], 2017. p. 32–37.

[18] LI, C.; JIANG, W.; ZOU, X. Botnet: Survey and case study. In: *2009 Fourth International Conference on Innovative Computing, Information and Control (ICICIC)*. [S.l.: s.n.], 2009. p. 1184–1187.

[19] COSTA, V. G. T. D. Uma abordagem baseada em chamadas de sistema para a detecção de botnets em dispositivos móveis.

[20] FEILY, M.; SHAHRESTANI, A.; RAMADASS, S. A survey of botnet and botnet detection. In: *2009 Third International Conference on Emerging Security Information, Systems and Technologies*. [S.l.: s.n.], 2009. p. 268–273.

[21] MEIDAN, Y. et al. N-baiot—network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, v. 17, n. 3, p. 12–22, 2018.

[22] ANTONAKAKIS, M. et al. Understanding the mirai botnet. In: *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, 2017. p. 1093–1110. ISBN 978-1-931971-40-9. Disponível em: <https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis>.

[23] BEZERRA, V. H. et al. One-class classification to detect botnets in iot devices. In: SBC. *Anais do XVIII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*. [S.l.], 2018. p. 43–56.

[24] JABEZ, J.; MUTHUKUMAR, B. Intrusion detection system (ids): Anomaly detection using outlier detection approach. *Procedia Computer Science*, Elsevier, v. 48, p. 338–346, 2015.

[25] JOSE, S. et al. A survey on anomaly based host intrusion detection system. In: IOP PUBLISHING. *Journal of Physics: Conference Series*. [S.l.], 2018. p. 012049.

[26] LIAO, H.-J. et al. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, Elsevier, v. 36, n. 1, p. 16–24, 2013.

[27] JAVAID, A. et al. A deep learning approach for network intrusion detection system. *Eai Endorsed Transactions on Security and Safety*, v. 3, n. 9, p. e2, 2016.

[28] NOVAES, M. P. et al. Adversarial deep learning approach detection and defense against ddos attacks in sdn environments. *Future Generation Computer Systems*, Elsevier, v. 125, p. 156–167, 2021.

[29] MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. *Sistemas inteligentes-Fundamentos e aplicações*, Manole Ltda, v. 1, n. 1, p. 32, 2003.

[30] LIU, H.; LANG, B. Machine learning and deep learning methods for intrusion detection systems: A survey. *Applied Sciences*, v. 9, n. 20, 2019. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/9/20/4396>.

[31] AMER, M.; GOLDSTEIN, M.; ABDENNADHER, S. Enhancing one-class support vector machines for unsupervised anomaly detection. In: *Proceedings of the ACM SIGKDD workshop on outlier detection and description*. [S.l.: s.n.], 2013. p. 8–15.

[32] WANG, S.-C. Artificial neural network. In: *Interdisciplinary computing in java programming*. [S.l.]: Springer, 2003. p. 81–100.

[33] MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943.

[34] FUKUSHIMA, K. Cognitron: A self-organizing multilayered neural network. *Biological cybernetics*, Springer, v. 20, n. 3, p. 121–136, 1975.

[35] CUN, Y. L.; FOGELMAN-SOULIÉ, F. Modèles connexionnistes de l'apprentissage. *Intellectica*, Persée-Portail des revues scientifiques en SHS, v. 2, n. 1, p. 114–143, 1987.

[36] ZHAI, J. et al. Autoencoder and its various variants. In: *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. [S.l.: s.n.], 2018. p. 415–419.

[37] SCHMIDHUBER, J. Deep learning in neural networks: An overview. *Neural networks*, Elsevier, v. 61, p. 85–117, 2015.

[38] LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *nature*, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015.

[39] PINAYA, W. H. L. et al. Autoencoders. In: *Machine learning*. [S.l.]: Elsevier, 2020. p. 193–208.

[40] SEWAK, M.; SAHAY, S. K.; RATHORE, H. An overview of deep learning architecture of deep neural networks and autoencoders. *Journal of Computational and Theoretical Nanoscience*, American Scientific Publishers, v. 17, n. 1, p. 182–188, 2020.

[41] TSCHANNEN, M.; BACHEM, O.; LUCIC, M. Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069*, 2018.

[42] KO, I.; CHAMBERS, D.; BARRETT, E. Adaptable feature-selecting and threshold-moving complete autoencoder for ddos flood attack mitigation. *Journal of Information Security and Applications*, Elsevier, v. 55, p. 102647, 2020.

[43] VU, L. et al. Learning latent distribution for distinguishing network traffic in intrusion detection system. In: IEEE. *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. [S.l.], 2019. p. 1–6.

[44] NICOLAU, M.; MCDERMOTT, J. et al. Learning neural representations for network anomaly detection. *IEEE transactions on cybernetics*, IEEE, v. 49, n. 8, p. 3074–3087, 2018.

[45] NGUYEN, V. Q. et al. Clustering-based deep autoencoders for network anomaly detection. In: SPRINGER. *International Conference on Future Data and Security Engineering*. [S.l.], 2020. p. 290–303.

[46] ANTHI, E. et al. Eclipseiot: A secure and adaptive hub for the internet of things. *Computers & Security*, Elsevier, v. 78, p. 477–490, 2018.

[47] GARCIA, S.; PARMISANO, A.; ERQUIAGA., M. J. *IOT-23 dataset: A labeled dataset of malware and benign IOT traffic.* Disponível em: <https://www.stratosphereips.org/datasets-iot23>.