UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

GUSTAVO AMARAL DUARTE REGO

USO DE APRENDIZADO DE MÁQUINA PARA DETECÇÃO DE ATAQUES DDOS

BAURU
Novembro/2023

GUSTAVO AMARAL DUARTE REGO

USO DE APRENDIZADO DE MÁQUINA PARA DETECÇÃO DE ATAQUES DDOS

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Faculdade de Ciências, Campus Bauru.
Orientador: Prof. Dr. Kelton Augusto Pontara da Costa

BAURU
Novembro/2023

Gustavo Amaral Duarte Rego

Uso De Aprendizado de Máquina Para Detecção De Ataques DDoS

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Prof. Dr. Kelton Augusto Pontara da Costa
Orientador
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação

Profa. Dra. Simone das Graças Domingues Prado
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação

Prof. Dr. João Paulo Papa
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação

Bauru, 16 de Novembro de 2023.

Agradecimentos

Agradeço primeiramente à minha família e o apoio imenso para chegar até aqui. Agradeço também aos amigos que fiz ao longo desses 4 anos de graduação. Obrigado Cassiano, Matheus, Marry, Toki, Zastim, Nick, Nih, Davizão, Dhiego, e tantos outros por fazerem parte dessa pequena temporada da vida (espero que sejam renovados para as próximas).

Agradeço também à minha namorada que, por mais difícil que tenha sido a jornada, sempre esteve ao meu lado, me apoiando e incentivando a sempre melhorar. Obrigado, Gatinha!

Agradeço especialmente aos meus pais por me ajudarem nessa jornada e apoiarem meus objetivos, me dando apoio e entendendo as minhas dores. Especialmente meu pai que, mesmo de longe, sempre me apoiou e me apoia para que siga meus sonhos.

Aos professores, agradeço em especial ao meu orientador, Prof Kelton, por ter me inserido mais a fundo em um assunto tão interessante e me mostrado aplicações de I.A que não havia pensado.

Agradeço aos membros de um grupo seleto de pessoas que se reúnem corriqueiramente no *Discord* (e à galera do SACI) para destilar entretenimento a todos, tornando assim, a rotina menos massante e mais alegre em meio às tarefas e prazos.

Por fim, agradeço aos meus gatos, às bandas *Ayreon* e *Epica*, ao Pinóquio de *Lies of P*, por me proporcionarem momentos mais tranquilos nos melhores e piores momentos.

Resumo

A Segurança Cibernética é um tema que possui muita relevância nos dias atuais, uma vez que a sociedade depende cada vez mais do mundo digital e de suas ferramentas. Ataques nesse meio são comuns de diversas formas, mas um dos mais ameaçadores para um serviço é o *Distributed Denial of Service*, ou, Ataque de negação de serviço distribuído, que tem por objetivo esgotar recursos do *host*. Aprendizado de Máquina pode ser definido como uma ramificação da inteligência artificial, que foca no uso de dados e algoritmos para imitar a forma como humanos aprendem. Por ser altamente eficiente no reconhecimento de padrões, é possível utilizar algoritmos desse tipo para auxiliar no reconhecimento de ataques cibernéticos. Nesse trabalho são abordados estudos e implementações sobre o uso de algoritmos de redes neurais, bem como o tratamento de dados para verificar sua acurácia e viabilidade utilizando conjuntos de dados de tráfego de rede.

Abstract

Cybersecurity is a highly relevant topic in today's world, given society's increasing dependence on the digital realm and its tools. Attacks in this domain manifest in various forms, with one of the most threatening being Distributed Denial of Service (DDoS) attacks, which aim to deplete a host's resources. Machine Learning can be defined as a branch of artificial intelligence that focuses on using data and algorithms to mimic how humans learn. Due to its high efficiency in pattern recognition, algorithms of this kind can be employed to aid in the detection of cyber attacks. This study explores studies and the implementation of neural network, in addition to data processing to assess their accuracy and feasibility using network traffic datasets.
Keywords:DDoS, AI, Machine Learning, Cyber Security.

Lista de figuras

Lista de quadros

Lista de abreviaturas e siglas

AM	*Aprendizado de Máquina*
CSV	*Comma Separatade Value*
DDoS	*Distributed Denial of Service*
IA	*Inteligência Artificial*
IP	*Protocolo de internet*
RNA	*Rede Neural Artificial*

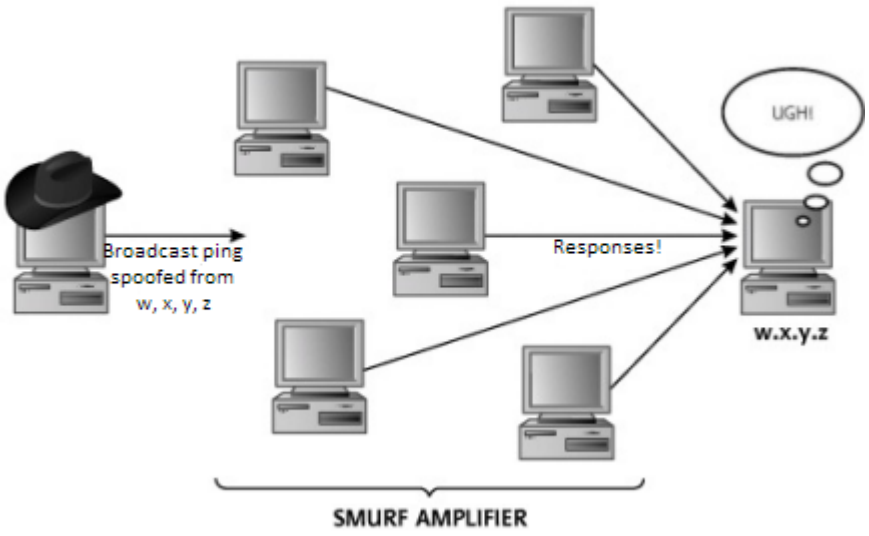
Sumário

1 Introdução

O mundo moderno e digitalizado é altamente dependente da internet e de serviços hospedados na rede. Isso faz com que certos serviços, por sua alta demanda e informações valiosas, sejam alvos de diversos tipos de ataques com diferentes objetivos. Seja para interromper seu funcionamento ou entrar na rede para coletar informações restritas.

Com isso em mente, um dos principais e mais comuns ataques são os chamados ataques *Distributed Denial of Service* (DDoS), que nada mais são do que tentativas de derrubar o serviço alvo. Isso pode ser feito de 2 formas. localmente, ou seja, o invasor precisa estar no local para efetuar o ataque, e de forma remota, no qual normalmente o invasor envia pacotes malformados ou até um número muito grande de pacotes para tentar derrubar o serviço esgotando recursos do sistema (SKOUDIS; LISTON, 2005) como ilustrado na Figura 1.

Figura 1 – Exemplo de Ataque DDoS em um servidor.



Fonte: Adaptada pelo autor de (SKOUDIS; LISTON, 2005)

Por esse motivo, é de extrema importância que tenham ferramentas que facilitem a detecção desse tipo de invasão. Existem duas principais maneiras de detectar esses ataques, sendo elas: análise *in-line* dos pacotes que chegam na rede em tempo real, e uma análise do fluxo de tráfico que é armazenado e posteriormente analisado com mais profundidade.

Nesse contexto, o uso de técnicas de Aprendizado de máquina para detecção de

ataques desse tipo recebeu muita atenção, visto que essas técnicas têm tido cada vez mais relevância nos dias atuais por permitirem prever uma grande gama de fenômenos e em alguns casos, pode produzir conhecimento com a interpretações de conjuntos de dados para uma conclusão de forma a atestar algo.(MURDOCH et al., 2019).

Para utilizar aprendizado de máquina a fim de detectar ataques, será utilizada a abordagem baseada em comportamento, ou seja, os algoritmos serão treinados para detectar anomalias no tráfego da rede que possa indicar um ataque DDoS. A técnica utilizada será a de *Deep Learning* utilizando a biblioteca *Tensorflow*, que consiste em uma técnica de aprendizado de máquina supervisioanda, onde o algoritmo será treinado para distinguir tráfego malicioso de tráfego comum utilizando um algoritmo de *Deep Learning*.

1.1 Problemática

Atualmente, ataques DDoS tem ficado mais intensos e difíceis de detectar, com a alta demanda da sociedade e de empresas em serviços de nuvem, cresce o número de vulnerabilidades presentes e, sendo assim, é necessária uma solução mais robusta para que esses serviços tenham uma segurança melhor. Existem soluções já prontas no mercado para a detecção desses ataques, porém, com o volume de tráfego crescendo na internet, será necessário criar uma solução mais eficiente, uma vez que as existentes exigem um alto poder computacional pela análise de pacotes ou análise de alto volume de dados de uma só vez.

Ainda que exista a possibilidade de produzir algoritmos mais eficientes para as soluções já existentes, esses algoritmos ainda demandariam um poder computacional maior do que um algoritmo SVM, uma vez que esse já está treinado e analisará pacotes e tráfego de forma muito mais eficiente e assertiva. Cabe, portanto, o desenvolvimento de aplicações e estudos nessa área, tornando essas soluções mais comuns.

1.2 Justificativa

De acordo com (ELIYAN; PIETRO, 2021), as soluções que são possíveis utilizando Aprendizado de Máquina possuem outros desafios que não são enfrentados hoje, como um alto custo e uma alta demanda de tempo incial para treinar os algoritmos e implementá-los, porém é importante destacar que essas soluções são mais eficientes e possuem um processo de decisão mais assertivo, a depender dos dados utilizados para o treinamento.

Diante do contexto exposto, justifica-se o desenvolvimento de uma aplicação que consiga, por meio de um algoritmo SVM treinado com um conjunto de dados rotulados,

detectar e separar um ataque DDoS de um tráfego normal na rede. A elaboração dessa aplicação levará em consideração a arquitetura de uma rede e a forma com que são feitos os ataques, utilizando-se de *datasets* específicos para esse fim e visará aumentar as chances de se mitigar ou evitar um ataque desse tipo.

1.3 Objetivo Geral

O objetivo principal desse trabalho é construir um algoritmo e uma aplicação que seja capaz de detectar ataques DDoS com base em técnicas de aprendizado de máquina com Redes Neurais Artificiais (RNAs).

1.4 Objetivos Específicos

- Fomentar o estudo nessa área que possui uma grande demanda de mercado, para que mais seja desenvolvido acerca do assunto.
- Estruturar um sistema capaz de detectar ataques que possa ser implementado e gere o resultado esperado, fazendo com que o poder computacional seja menor.
- Realizar comparações para que seja atestado qual a eficiência de modelos de aprendizado de máquina frente aos modelos tradicionais.

1.5 Organização do Trabalho

O presente trabalho está estruturado da seguinte forma: no Capítulo 2 está presente a fundamentação teórica abrangendo conceitos de ataques DDoS e Aprendizado de Máquina (AM). Já o Capítulo 3 diz respeito a como foi feito o modelo de treinamento, quais foram as ferramentas utilizadas e explicando melhor os parâmetros utilizados no treinamento e quais datasets foram utilizados para esse treinamento. No Capítulo 4, são descritos e analisados os dados gerados pelo modelo. Por fim, no Capítulo 5 são apresentadas as considerações finais sobre o trabalho e algumas possibilidades de trabalhos futuros.

2 Fundamentação Teórica

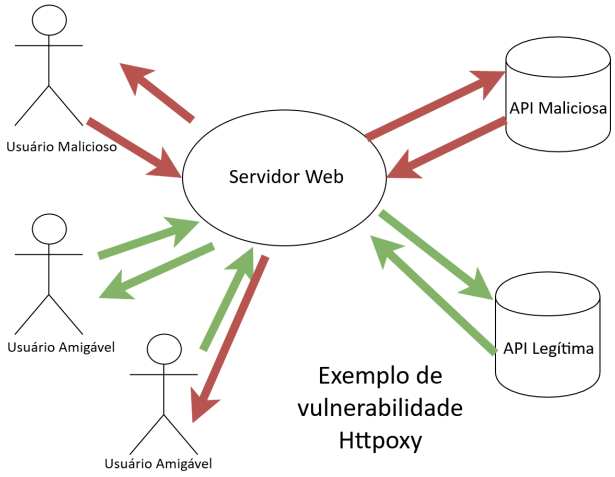
Neste capítulo, os conceitos mais importantes serão apresentados utilizados nesse trabalho para um melhor entendimento, tais como conceitos de aprendizado de máquina, bibliotecas e segurança cibernética (mais especificamente, ataques DDoS).

2.1 Cibersegurança

A cibersegurança é uma área de extrema importância para empresas e pessoas, uma vez que, em sua maioria, armazenam suas informações em ambientes digitais e atualmente, a demanda por serviços de internet para tal finalidade está aumentando significativamente com a popularização da computação em nuvem. O objetivo da cibersegurança é proteger chamados "ativos", que podem ser *hardware*, *software* ou dados em si. De acordo com (CHRISTEN; GORDIJN; LOI, 2020), a cibersegurança pode ser dividida em dois grandes campos: Segurança da informação e Segurança de sistema.

Para o estudo apresentado, será abordado o segundo campo, e no caso, os sistemas-alvo desse estudo são sistemas que não necessariamente possuem dados, mas seu funcionamento é necessário para o fluxo correto de uma tarefa, como o *login* em um sistema, por exemplo. Caso o invasor consiga uma brecha em algum fluxo crítico do sistema, poderá explorá-la para comprometer seu funcionamento, ou até mesmo impedir a conexão do cliente com o servidor.

Figura 2 – Exemplo de vulnerabilidade Httpoxy



Fonte: Adaptada de https://symfony.fi/page/httpoxy-exploit-example

Um exemplo de falha pode ser representado na Figura 2, que demonstra como um usuário malicioso consegue, por meio de uma vulnerabilidade no protocolo HTTP, enviar requisições falsas ao servidor utilizando a conexão de um usuário já conectado, e assim, redirecionar as repostas de um servidor malicioso controlado pelo atacante para o usuário por meio do servidor confiável. Isso faz com que o usuário malicioso tenha controle sobre as respostas que o outro usuário está recebendo, e possa enviar informações ou pacotes maliciosos a fim de coletar informações sigilosas.

Nesse trabalho, porém, é abordado o ataque DDoS, que consiste no atacante utilizar-se de múltiplos clientes para conectar-se em um único servidor com o objetivo de esgotar seus recursos, podendo ser recursos de rede e conexão ou recursos de *hardware*, como memória ou processamento

Um dos maiores desafios de se implementar de forma eficiente um sistema de detecção de ataques DDoS é a necessidade de um conjunto de dados expressivo para o treino da rede neural e também o cuidado com a privacidade dos dados e o viés algorítmico.

2.1.1 Ataques DDoS

Segundo (HOQUE; BHATTACHARYYA; KALITA, 2015), um ataque DDoS é um ataque coordenado gerado usando muitos *hosts* comprometidos. Inicialmente, um atacante identifica vulnerabilidades em uma rede para instalar programas maliciosos em várias máquinas e assumir o controle delas. Em seguida, o atacante utiliza esses *hosts* comprometidos para enviar pacotes de ataque à vítima sem esse conhecimento. Dependendo da intensidade dos pacotes de ataque e do número de *hosts* usados para o ataque, ocorrem danos proporcionais na rede da vítima. Se o número de *hosts* comprometidos for muito grande, isso pode interromper uma rede ou um servidor web em um período muito curto de tempo.

O objetivo de um atacante DDoS é interromper uma rede de forma que ela não consiga fornecer serviços aos usuários legítimos. Para lançar um ataque, um atacante geralmente segue quatro etapas básicas:

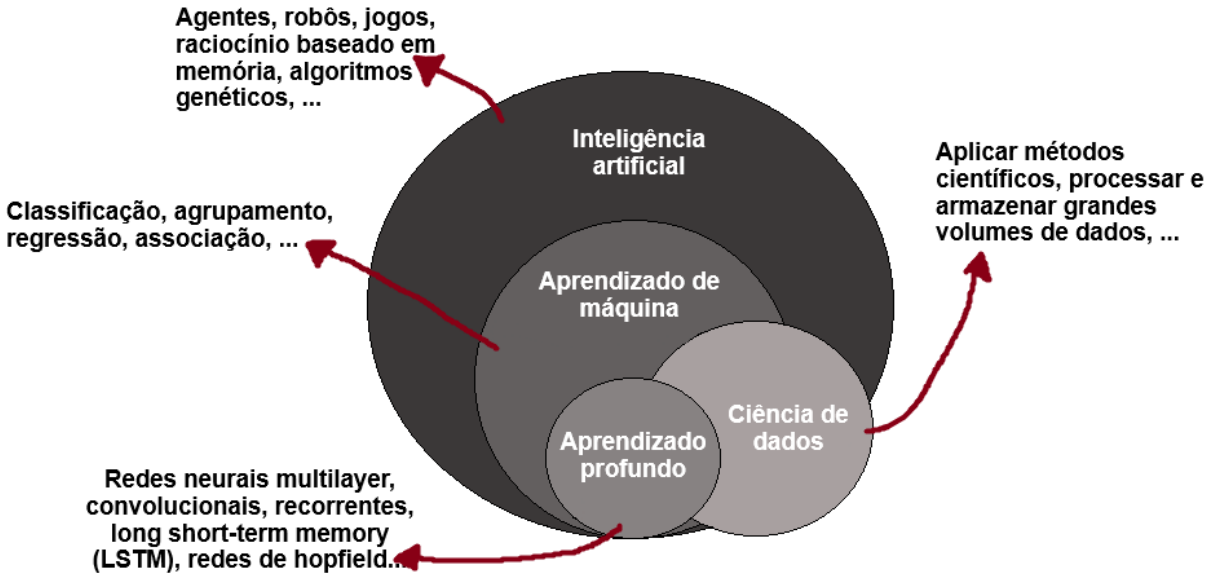
- Coleta de informações para escanear uma rede e encontrar *hosts* vulneráveis para usá-los posteriormente no ataque;
- Comprometimento dos *hosts* para instalar *software* mal intencionado ou programas maliciosos nos *hosts* comprometidos (chamados de zumbis) para que possam ser controlados apenas pelo atacante;
- Lançamento do ataque para comandar os zumbis a enviar pacotes de ataque com intensidades especificadas à vítima; e

- Limpeza para remover todos os registros ou arquivos de histórico da memória.

2.2 Aprendizado de Máquina

Segundo (FRADKOV, 2020)conceito principal de AM é uma área da Inteligência Artificial que tem como objetivo o desenvolvimento de algoritmos capazes de adquirir conhecimentos de forma automática e acumulá-los através de soluções bem sucedidas de problemas anteriores. Essa área começou com estudos do psicólogo Frank Rosenblat, da Universidade de Cornell, em Nova York, que se baseou em um modelo real de neurônio humano. Alguns modelos apresentam formas e métodos diferentes para serem treinados e executados, modificando assim seu funcionamento e a sua eficiência. Atualmente, há um amplo estudo na área de AM para tornar mais eficientes esses algoritmos.

Figura 3 – Diagrama de Venn de Inteligência Artificial.



Fonte: Adaptado de serpro.gov.br[1]

É importante ressaltar também que AM é uma sub-área da IA como um todo, como ilustrado na Figura 3, sendo somente uma ferramenta muito útil para fazermos análises de conjuntos extensos de dados. Essa área possui um potencial enorme no futuro, uma vez que atualmente há estudos na área de IA generativa e existem inúmeras aplicações para a IA utilizando AM como base, como o trabalho exposto, detecção de intrusão, checar padrões em casos de *phishing* na internet, entre outros.

Há uma série de algoritmos dentro da área de AM que podemos utilizar para detectar ataques DDoS, como *Support Vector Machine*, RNAs, que podem ser treinados

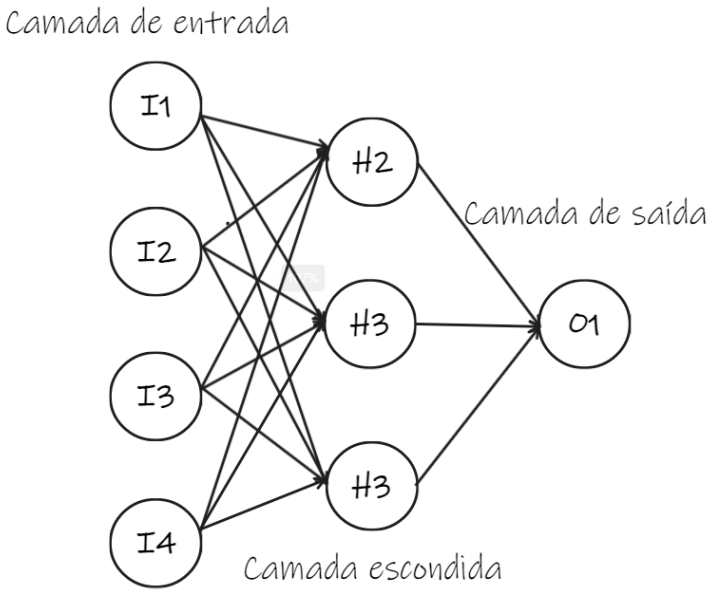
[1] *Fonte: https://www.serpro.gov.br/menu/noticias/noticias-2019/democratizando-a-inteligencia-artificial*

usando dados rotulados para detectar padrões associados aos ataques. Além disso, outras técnicas de aprendizado não supervisionado, como algoritmos de agrupamento (*Clustering*) podem ser utilizadas para encontrar comportamentos anômalos na rede.

2.2.1 Rede Neural Artificial

De acordo com (ANITESCU et al., 2019), RNA é um algoritmo que tenta imitar a forma como humanos aprendem, por meio de um dado de entrada e um dado de saía. Os neurônios, ou nós, da cadeia são conectados formando um grafo, que pode possuir n camadas, sendo que cada neurônio possui uma entrada e uma saída, para a próxima camada, formando um fluxo, como pode ser visualizado na Figura 4. Esses modelos aprendem de forma recorrente de acordo com os parâmetros escolhidos inicialmente para o reconhecimento do padrão a ser aprendido. Tais modelos podem ser facilmente construídos utilizando bibliotecas da linguagem Python, como o TensorFlow, que é abordado na seção 2.3.

Figura 4 – Exemplo de funcionamento de uma RNA



Fonte: Elaborado pelo autor

2.3 TensorFlow

De acordo com a página oficial (ABADI et al., 2015) o Tensorflow é um sistema de aprendizado de máquina de larga escala que utiliza grafos de fluxo de dados para representar operações e estados compartilhados. Pode ser executado em várias máquinas e tipos de dispositivos computacional, como processadores, processadores Gráficos e processadores *Tensor*, que é uma arquitetura específica para utilização em IA. Isso

oferece flexibilidade aos desenvolvedores e os permitem experimentar em diferentes ambientes, com diferentes recursos e verificar suas diferenças de desempenho.

A escolha da biblioteca Tensorflow se deu por conta da facilidade com que é possível construir, modificar e utilizar um modelo RNA, extensa documentação, uma comunidade bem ativa, integração com praticamente qualquer tipo de processador e uma ampla gama de funções disponíveis para diversos casos de uso. Foi escolhido, especificamente um modelo RNA (*Keras Sequential*) por ser simples de implementar e por ser uma rede neural profunda, é extremamente poderoso e parametrizável com a quantidade de dados utilizada.

2.4 *Pandas*

Pandas, em sua versão 2.1.1, é a biblioteca escolhida para o tratamento e a manipulação dos dados do conjunto de dados. Essa escolha se deu pela facilidade com que a biblioteca consegue ler arquivos do tipo *comma separatade value (CSV)* e convertê-los para um quadro de dados que pode ser compreendido e utilizado pela linguagem *Python*. Outro motivo da escolha da biblioteca é porque possui uma ampla documentação e uma comunidade ativa de usuários, o que torna mais rápido o desenvolvimento de ferramentas com essa biblioteca. Além disso, possui integração com ferramentas de visualização como o *matplotlib*, biblioteca também utilizada para gerar os gráficos nesse trabalho.

3 Materiais e Métodos

Este capítulo se trata do processo de desenvolvimento e treinamento do modelo de AM baseado em uma RNA para a detecção de ataques DDoS, que envolve a coleta do conjunto de dados, o desenvolvimento do modelo, o treinamento desse modelo. Além disso, é importante seguir algumas diretrizes de ética quando se trabalha com algoritmos de IA, como por exemplo, a privacidade dos dados, e por esse motivo, o conjunto de dados utilizado, (DEVENDRA416, 2020), é público e foi tratado para que não haja nenhum tipo de viés no treinamento da rede.

3.1 Materiais Gerais

Foram escolhidas as seguintes ferramentas para a execução do projeto:

- **Sistemas Operacionais**: Windows 11;
- **Ambiente de Desenvolvimento Integrado**: Microsoft Visual Studio Code e Google Collab;
- **Linguagem de Programação**: *Python*; e
- **Ferramentas**: *Jupyter*, *Google Collab*, *Pandas* e TensorFlow

Hardware

O Hardware utilizado está presente no Quadro 1 abaixo:

Quadro 1 – Especificações do notebook usado

Marca	Lenovo
Modelo	Ideapad Gaming 3i i5-10300H
Memória RAM	32GB
Armazenamento	1TB SSD
Placa de Vídeo	NVIDIA GeForce GTX 1650 com 4GB

Fonte: Elaborado pelo autor.

Os materiais gerais do projeto serão descritos a seguir, como os dados utilizados, TensorFlow e o *Pandas*.

3.1.1 Dados

Os dados utilizados foram retirados do conjunto de dados *"DDoS DataSet"* (DEVENDRA416, 2020), que possui 12.794.627 de dados rotulados como sendo um ataque DDoS ou tráfego benigno, com 84 parâmetros. O tratamento feito nos dados foi basicamente de retirar os dados irrelevantes, ruídos presentes nos dados e escolha das características a serem utilizadas no treinamento.

3.2 Tratamento dos dados

Esta seção apresenta o processo de escolha e tratamento dos dados (DEVENDRA416, 2020) para o treinamento da rede neural. Para esse tratamento foi utilizada a biblioteca *Pandas* já descrita acima.

Inicialmente, foi necessária a seleção das características que serão utilizadas para o treinamento. As características selecionadas foram:

- Pacotes por Segundo eepresenta a quantidade de pacotes que está fluindo na rede naquele segundo. Esse dado é relevante devido à natureza de ataques DDoS, que envolve muitas requisições para um mesmo destino em um curto período de tempo;
- Tamanho médio dos pacotes representa o tamanho médio dos pacotes no mesmo período de tempo. Essa característica se torna relevante em um reconhecimento pois para um ataque DDoS, é necessária uma quantidade grande de pacotes na rede e, quanto menores são os pacotes, mais rápida ocorre a transmissão, e por consequência, maior o fluxo de pacotes na rede;
- Endereço de IP (Protocolo de Internet, do inglês) de destino representa o endereço de IP de destino do pacote analisado. Tal informação é relevante pois, se há uma quantidade incomumente grande de pacotes para um mesmo destino, é possível que se configure um ataque DDoS, uma vez que é possível congestionar o tráfego de rede com a quantidade de pacotes para um único *host*; e
- Rótulo indica se o conjunto de parâmetros na linha é um conjunto rotulado como um ataque DDoS ou um tráfego de rede comum.

Após essa seleção de características, podendo ser visualizadas na Tabela 1, foi feito um tratamento na quantidade de dados no conjunto de dados pois assim, é possível comparar como o modelo se comporta com quantidades diferentes das amostras. Isso é importante pois, de acordo com (YARALLY et al., 2023) há um alto

consumo de energia por parte de modelos de IA sendo treinados, e, é necessária uma ação para diminuir tal consumo, sendo o tempo de treinamento algo relevante para tal.

Tabela 1 – Amostra de dados utilizados

DADOS				
índice	Pacotes p/ Segundo	Tamanho médio dos pacotes	IPs de destino	Rótulo
0	18.365417466065487	809.4189189189186	14620	ddos
1	0.024062716293741	0.0	34286	ddos
2	0.0999270532511266	0.0	32128	ddos
3	33.06503419869252	76.75	31540	ddos
4	33.253682845375124	76.75	31540	ddos

Fonte: Elaborado pelo autor

3.3 Métodos

Nesta seção, são apresentados como foi feito o modelo, análises mais técnicas acerca dos dados utilizados e a forma com que o modelo foi treinado. São mencionados também as dificuldades e desafios durante sua elaboração.

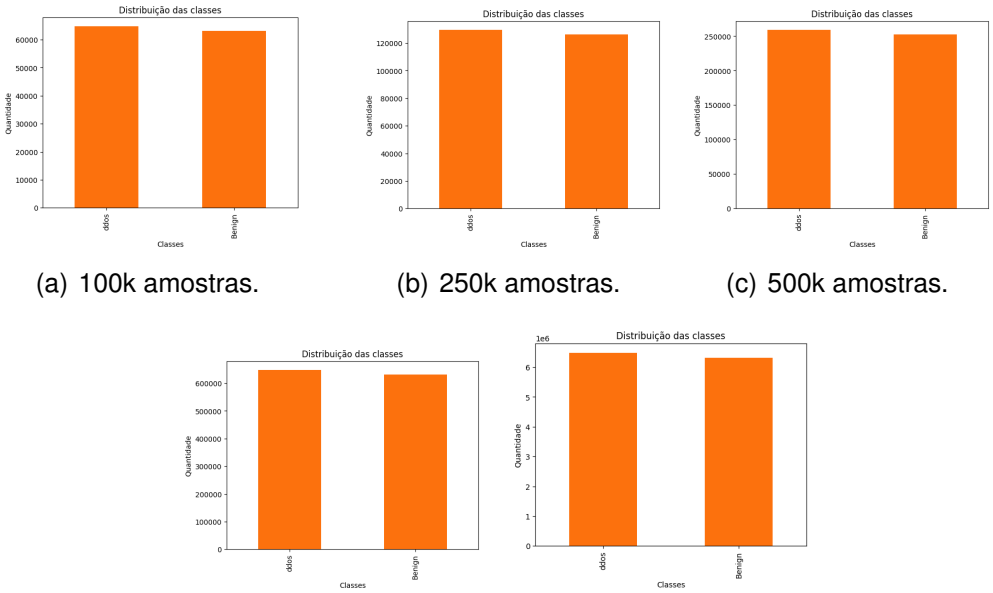
3.3.1 Separação dos dados

Os dados apresentados no início dessa seção foram separados em cinco diferentes conjuntos de dados e salvos em diferentes arquivos, tais arquivos *.CSV* são formados por n amostras dos dados totais, sendo 100 mil, 250 mil, 500 mil, 1 milhão e 10 milhões de valores com as quatro características:

- **ProcessedDataset100k.csv**: Conjunto de dados com 100.000 amostras;
- **ProcessedDataset250k.csv**: Conjunto de dados com 250.000 amostras;
- **ProcessedDataset500k.csv**: Conjunto de dados com 500.000 amostras;
- **ProcessedDataset1M.csv**: Conjunto de dados com 1.000.000 de amostras; e
- **ProcessedDataset10M.csv**: Conjunto de dados com 10.000.000 amostras.

Tais dados foram tratados utilizando funções da biblioteca *Pandas*. Inicialmente, foi necessário montar um quadro de dados para que pudessem ser lidos pelo *Pandas* e *Numpy*. Feito isso, foi checado também a quantidade de cada tipo de dados (benigno ou não) em cada conjunto de dado feito. Pode-se verificar isso na figura 5.

Figura 5 – Gráficos com o número de amostras em cada conjunto



(a) 100k amostras. (b) 250k amostras. (c) 500k amostras.

(d) 1M amostras. (e) 10M amostras.

Fonte: Elaborada pelo autor

Após a separação, foram feitos os quadros de dados utilizando a biblioteca *Pandas*, para que o *Python* enxergasse os dados. Esses dados já estão representados na Tabela 1, e foram feitos utilizando a função *pd.DataFrame* para transformar os dados do arquivo *".CSV"* no quadro de dados, como pode ser visualizado no Código 1 e Código 2.

```
1 dataSet = pd.read_csv("DDoS_Dup.csv")
2 df = pd.DataFrame(dataSet)
```

Código 1 – Transforma o conjunto de dados em DataFrame

```
1  # Modificando o dataset
2  divisor = 100
3  avaragePacketSize = df["Pkt Len Mean"][::divisor]
4  packetsPerSecond = df["Flow Pkts/s"][::divisor]
5  label = df["Label"][::divisor]
6  df.replace([np.inf, -np.inf], 0, inplace=True)
7  x = df["destip_unique"].max()
8  destinationIP = df["destip_unique"]
9  destinationIP = destinationIP
10 destinationIP = destinationIP[::divisor]
```

Código 2 – Modifica o DataFrame para gerar conjuntos de dados diferentes

Após a confecção do quadro de dados, foi feito também a substituição de valores infinitos nos dados, vistos na linha 6 do Código 2. Esses valores, para serem lidos pelo modelo, precisam ser um valor inteiro ou ponto flutuante, portanto foram substituídos

por 0. Após isso, foi feita uma conversão nos endereços IP do quadro de dados, uma vez que estavam em formato de texto e devem ser lidos como inteiros ou reais. Essa conversão pode ser vista no Código 4, na qual todos os valores foram convertidos para inteiro.

```
destip_unique = []
destip_unique = df["Dst IP"].unique()
le = preprocessing.LabelEncoder()
df["destip_unique"] = le.fit_transform(df["Dst IP"])
```

Código 3 – Transformação de número de IPs

```
x = np.array(df1[["Destination IP", "Avarage Packet Size", "Packets Per
    Second"]])
y = np.array(df1["Label"])
scaler = StandardScaler()
X_scaled = scaler.fit_transform(x)
label_mapping = {"ddos": 1, "Benign": 0}
label_to_number = np.vectorize(lambda x: label_mapping[x])
y = label_to_number(y)
```

Código 4 – Transformação dos dados

Por fim, os 4 parâmetros (Pacotes p/Segundo, Tamanho médio dos pacotes IPs de destino e Rótulo) foram linearizados para alimentar o modelo de treinamento, podendo ser visualizados no Código 3, os rótulos foram transformados em inteiros e foram criados dois vetores, um com o conjunto de parâmetros utilizados para o reconhecimento e outro com o rótulo. Após a linearização, os dados foram separados em treino, teste e validação, sendo 30% para treino, 70% para teste e 14% para validação. O Código pode ser visualizado no Código 5

```
X_temp, X_test, Y_temp, Y_test = train_test_split(
    X_scaled, y, test_size=0.3, random_state=42
)
X_train, X_val, Y_train, Y_val = train_test_split(
    X_temp, Y_temp, test_size=0.2, random_state=42
)
```

Código 5 – Transformação de número de IPs

3.3.2 Desenvolvimento do modelo

Já tendo os dados que serão utilizados, o primeiro passo para o desenvolvimento da RNA que será treinada é necessária a instalação da biblioteca TensorFlow. Uma vez que o TensorFlow é um pacote relativamente conhecido, foi utilizado o tutoria de instalação expostos por (ABADI et al., 2015) na documentação oficial.

Para o desenvolvimento do modelo de RNA foi escolhido o *keras.sequential* por ser uma classe que possibilita criar modelos camada por camada de maneira simples e intuitiva. No modelo construído foram utilizadas duas camadas do tipo *"Rectified Linear Unit"* (*ReLU*), uma função de ativação não linear que não utiliza funções exponenciais no seu funcionamento, o que a torna uma função computacionalmente mais leve. No modelo construído, froam utilizadas duas camadas ReLU com 64 neurônios na primeira camada e 32 na segunda. Além das duas camadas ReLU, há uma última camada do tipo *sigmoid* que serve para transformar em binária a saída da rede neural, ou seja, a depender do valor enviado pela última camada ReLU, a saída se dá como sendo um ataque ou um tráfego comum. Esse desenvolvimento pode ser visualizado no Código 6.

```
1 destip_unique = []
2 destip_unique = df["Dst IP"].unique()
3 le = preprocessing.LabelEncoder()
4 df["destip_unique"] = le.fit_transform(df["Dst IP"])
```

Código 6 – Transformação de número de IPs

O modelo, depois de desenvolvido, foi também otimizado utilizando o algoritmo *Adam*, responsável por atualizar os pesos da rede neural durante seu funcionamento. Foi utilizado também uma função de perda *binary_entropy* que é responsável por avaliar o modelo caso esteja cometendo muitos erros na classificação.

3.3.3 Treinamento do modelo

Para os parâmetros de treinamento do modelo, são necessários dois importantes fatores:

- **Lote**: Representa o a quantidade de dados que serão enviados por vez ao modelo para treina-lo. Esse parâmetro é essencial para o treinamento da rede, sendo que quanto menor o número, mais atualizações de parâmetros a rede fará ao longo da execução. Isso faz com que o resultado convirja mais rápido, porém, caso o tamanho do lote seja muito pequeno, podem haver atualizações de pesos muito voláteis; e
- **Época**: Número de vezes que o conjunto de dados passa pela rede neural, para treina-la. A cada época, o modelo calcula gradientes e otimiza-os com base no algoritmo de otimização utilizado (no caso, *binary_entropy*). Algo a se observar é que é necessário escolher um número adequado de épocas, a depender do seu conjunto de dados, para treinar o modelo, uma vez que é possível que o modelo não tenha tempo suficiente para aprender caso seja um número muito pequeno de épocas.

O treinamento do modelo foi feito com os 5 conjuntos de dados apresentados na subseção 3.3.1, com 10 épocas e alimentando os dados em lotes de 32 dados por vez. O exemplo pode ser visto no Código 7.

```
model = tf.keras.Sequential(
    [
        tf.keras.layers.Dense(64, activation="relu", input_shape=(X_train.shape[1],)),
        tf.keras.layers.Dense(32, activation="relu"),
        tf.keras.layers.Dense(
            1, activation="sigmoid"
        ),  # Esse output para o sigmoid é necessário para binarizar a saída
    ]
)
```

Código 7 – Treinamento do modelo

Por fim, para avaliar a acurácia do modelo de classificação das características, foi utilizada a função *model.predict* que, com os dados de teste como entrada, retorna um array NumPy com números entre 0 e 1 representando a probabilidade de uma instância pertencer à classe correta. O desenvolvimento pode ser visto no Código 8.

```
# avalia o modelo no conjunto de testes
Y_pred = model.predict(X_test)
Y_pred = (Y_pred >= 0.5).astype(int)
```

Código 8 – Avaliação do modelo

4 Análise dos Resultados

Neste capítulo é feita uma análise dos resultados obtidos com o modelo treinado, destacando-se o a perda (*loss*) versus a acurácia (*accuracy*) do modelo.

4.1 Resultados Gerados

Os resultados obtidos com o treinamento da RNA são analisados através da confecção de dois gráficos relacionando acurácia e perda com o número de épocas para cada conjunto de dados testados no treino.

Todos os gráficos nessa seção foram gerados com o uso da biblioteca *matplotlib*, uma biblioteca da linguagem *Python* que permite montar gráficos a partir de dados de diversas fontes. Os dados dessa seção foram extraídos dos resultados obtidos com o treinamento do modelo desenvolvido na subseção 3.3.2. Tais dados foram armazenados em arquivos *.CSV* para uso futuro. Um exemplo de como os gráficos foram esboçados pode ser visualizado no Código 9

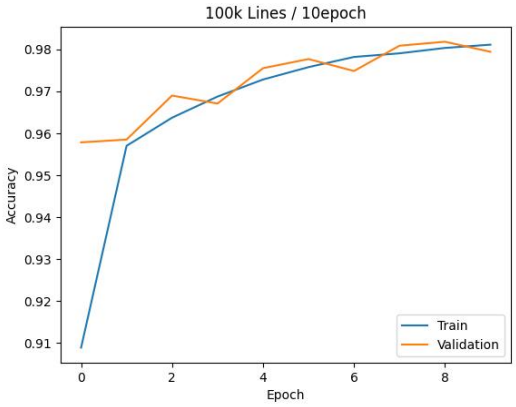
```
# Accuracy
plt.plot(history.history["accuracy"])
plt.plot(history.history["val_accuracy"])
plt.title("500K 20 Epochs Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend(["Train", "Validation"], loc="lower right")
plt.show()
```

Código 9 – Gera o gráfico.

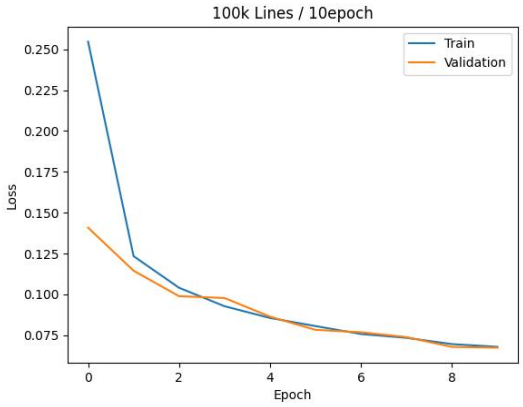
4.1.1 Gráficos

Os gráficos gerados serão apresentados nessa seção e analisados na subseção 4.1.2 com base na comparação entre os conjuntos de dados.

Figura 6 – Gráficos com 100k amostras

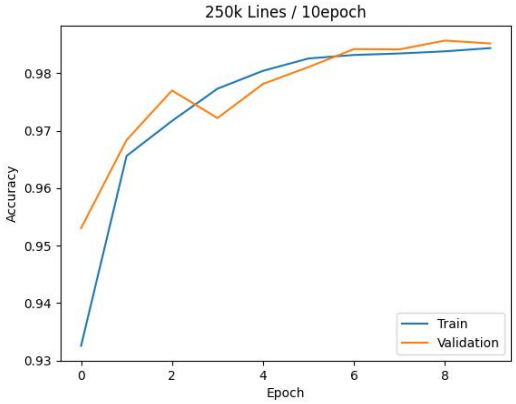


(a) Acurácia

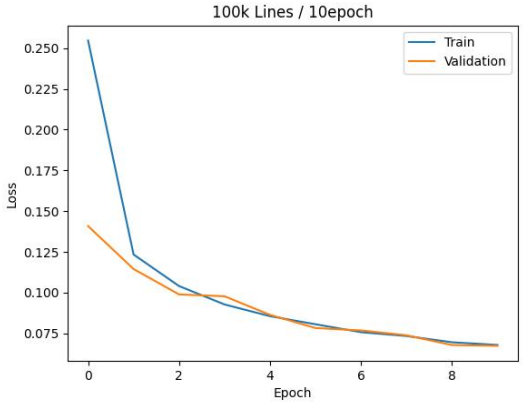


(b) Perda

Figura 7 – Gráficos com 250k amostras

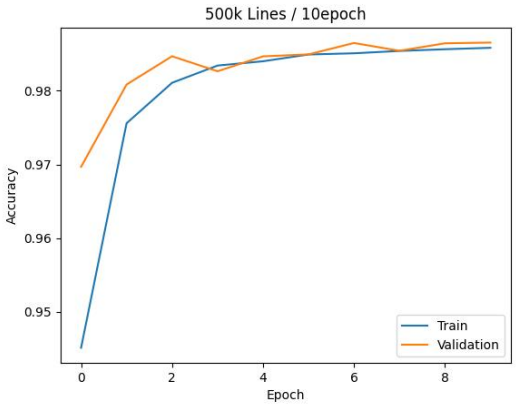


(a) Acurácia

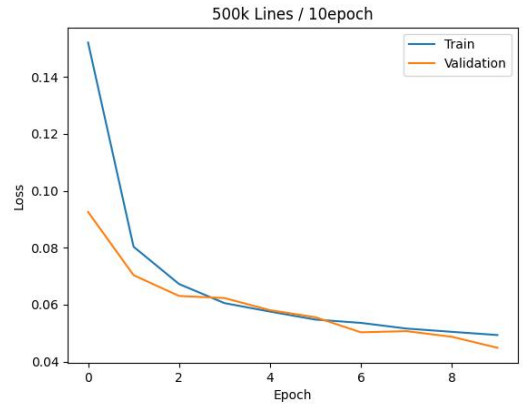


(b) Perda

Figura 8 – Gráficos com 500k amostras

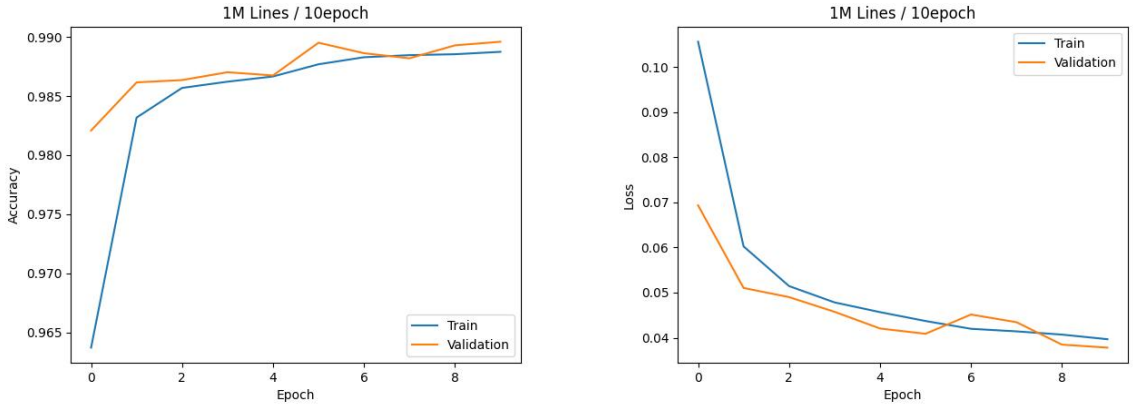


(a) Acurácia



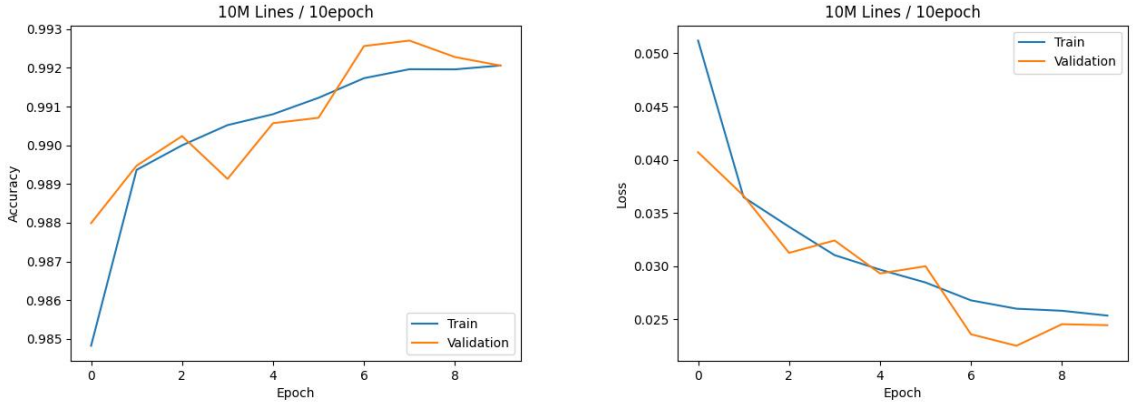
(b) Perda

Figura 9 – Gráficos com 1M amostras



(a) Acurácia (b) Perda

Figura 10 – Gráficos com 10M amostras

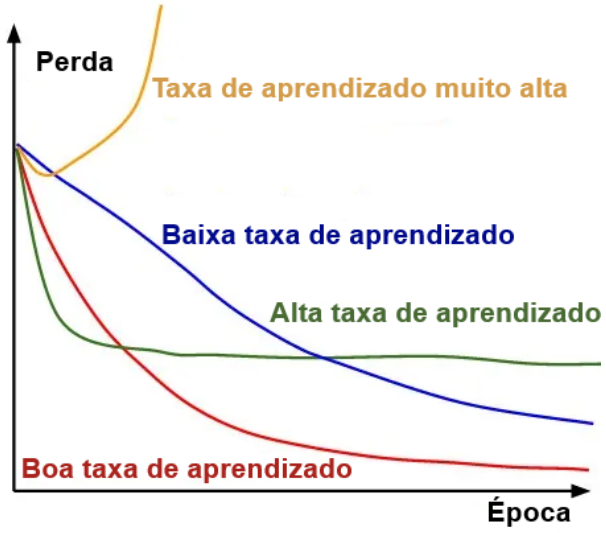


(a) Acurácia (b) Perda

4.1.2 Análise

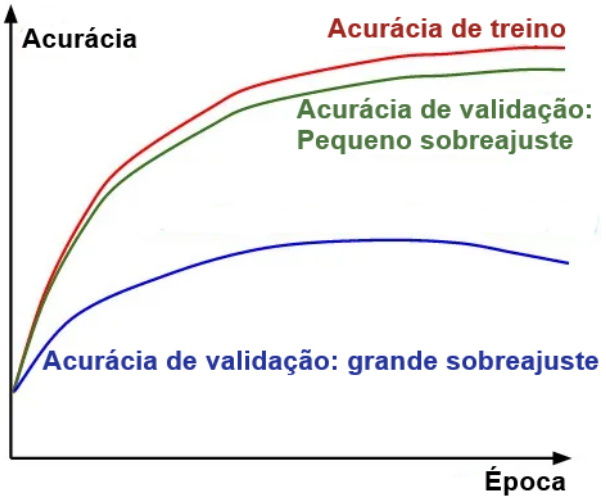
Para que seja feita uma análise correta dos gráficos, devemos ter como base uma curva ideal de aprendizado e uma curva ideal de perda do modelo. Segundo (SHEN et al., 2020) a curva ideal pode ser representada na Figura 11 e na Figura 12. Quanto maior a derivada da curva curva de acurácia, mais rápido o modelo aprende, sendo que o ideal é a presença de uma curva logarítmica em ambos os gráficos.

Figura 11 – Curva ideal de acurácia de um modelo RNA



Fonte: (SHEN et al., 2020)

Figura 12 – Curva ideal de perda de um modelo RNA



Fonte: (SHEN et al., 2020)

Com base nos dados coletados nos cinco experimentos, é possível inferir que em alguns casos são necessárias mais épocas para treinar melhor o modelo, como é o caso do gráfico das figuras 7(a) e 7(b), uma vez que a acurácia e a perda dos outros gráficos estão melhores. Tal comportamento é esperado em um modelo RNA, uma vez que quanto mais dados são utilizados como entrada no modelo, mais acurado é o resultado. O que pode-se perceber nos resultados também é que, mesmo com um número menor de dados, visto nas figuras 8(a) e 8(b), é possível inferir que os resultados são satisfatórios, uma vez que tanto a tendência de perda quanto a tendência de acurácia aumentam em uma taxa parecida com a logarítmica.

Ademais, é possível fazer a análise com base na comparação das curvas de teste com a de validação. Com relação a todos os casos, a curva de teste segue a tendência da curva de validação; porém, há uma curva que segue com mais precisão, dada pelas 100.000 amostras. Por esse motivo, é inferido de que é possível treinar uma rede *ReLU* com menos dados e obter uma acurácia próxima da ideal, uma vez que com a amostra de 10.000.000 de dados, dada pela figura 11(b), possui o valor ótimo entre 0.992 e 0.993 na 10° época, e com 100.000 dados, na 10° época, encontra valores próximos de 0.98, o que representa uma diferença de 1% entre as curvas. Contudo, para atingir este resultado, é necessário aumentar em 100 vezes a quantidade de dados, o que pode ser inviável em outros cenários.

5 Considerações Finais

O objetivo principal desse trabalho foi construir um algoritmo capaz de detectar ataques DDoS com base em técnicas de AM com RNAs. Especificamente, os estudos foram direcionados a estruturar um algoritmo e um modelo que fosse capaz de detectar ataques e que possa ser implementado em um sistema, uma vez que representa uma solução viável. Diante disso, optou-se pelo uso de ferramentas que auxiliam no desenvolvimento de modelos de RNA para que fosse mais acessível a quem não necessariamente sabe o funcionamento nuclear de RNAs, uma vez que tais ferramentas são intuitivas e possuem uma ampla documentação.

Diante dessa implementação, foi possível perceber que os resultados dentro do mesmo conjunto de dados, ainda que relativamente diferentes, foram parecidos e entregaram o que se esperava de um modelo de AM e que é necessário um desenvolvimento maior acerca do tópico utilizando outras técnicas.

Contudo, não foi possível fazer a comparação com outras técnicas devido à dificuldade de encontrar dados para o treino de diferentes modelos não supervisionados e de *Support Vector Machine*. Foi observado, também, que não necessariamente quando se aumenta a quantidade de dados os benefícios crescem linearmente. Alguns modelos necessitam de menos dados e podem ser treinados de maneira eficiente em menos épocas, a depender dos parâmetros escolhidos.

5.1 Trabalhos Futuros

Considerando uma continuidade futura do trabalho, podemos explorar alguns aspectos da pesquisa:

- Utilização de outras técnicas de AM para comparação;
- Modificar os parâmetros selecionados para avaliar como o modelo se ajusta a diferentes parâmetros;
- Modificação dos hiper-parâmetros do modelo, como número de neurônios, camadas e diferentes funções de ativação;
- Utilizar outros conjuntos de dados com o modelo para confirmar seu funcionamento; e
- Expandir o escopo do modelo para reconhecimento de outros ataques cibernéticos.

Referências

ABADI, M.; AGARWAL, A.; BARHAM, P.; BREVDO, E.; CHEN, Z.; CITRO, C.; CORRADO, G. S.; DAVIS, A.; DEAN, J.; DEVIN, M.; GHEMAWAT, S.; GOODFELLOW, I.; HARP, A.; IRVING, G.; ISARD, M.; JIA, Y.; JOZEFOWICZ, R.; KAISER, L.; KUDLUR, M.; LEVENBERG, J.; MANé, D.; MONGA, R.; MOORE, S.; MURRAY, D.; OLAH, C.; SCHUSTER, M.; SHLENS, J.; STEINER, B.; SUTSKEVER, I.; TALWAR, K.; TUCKER, P.; VANHOUCKE, V.; VASUDEVAN, V.; VIéGAS, F.; VINYALS, O.; WARDEN, P.; WATTENBERG, M.; WICKE, M.; YU, Y.; ZHENG, X. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Software available from tensorflow.org. Disponível em: https://www.tensorflow.org/. Acesso em: 25 out 2023.

ANITESCU, C.; ATROSHCHENKO, E.; ALAJLAN, N.; RABCZUK, T. Artificial neural network methods for the solution of second order boundary value problems. *Computers, Materials & Continua*, v. 59, n. 1, 2019.

CHRISTEN, M.; GORDIJN, B.; LOI, M. *The ethics of cybersecurity*. Springer Nature, 2020. Disponível em: https://link.springer.com/book/10.1007/978-3-030-29053-5. Acesso em: 18 out. 2023.

DEVENDRA416. *DDoS Dataset*. 2020. Disponível em: https://www.kaggle.com/datasets/devendra416/ddos-datasets/data. Acesso em: 26 jun 2023.

ELIYAN, L. F.; PIETRO, R. D. Dos and ddos attacks in software defined networks: A survey of existing solutions and research challenges. *Future Generation Computer Systems*, Elsevier, v. 122, p. 149–171, 2021. Disponível em: https://www.sciencedirect.com/science/article/pii/S0167739X21000911. Acesso em: 17 set. 2023.

FRADKOV, A. L. Early history of machine learning. *IFAC-PapersOnLine*, Elsevier, v. 53, n. 2, p. 1385–1390, 2020. Disponível em: https://www.sciencedirect.com/science/article/pii/S2405896320325027. Acesso em: 12 out. 2023.

HOQUE, N.; BHATTACHARYYA, D. K.; KALITA, J. K. Botnet in ddos attacks: Trends and challenges. *IEEE Communications Surveys Tutorials*, v. 17, n. 4, p. 2242–2270, 2015. Disponível em: https://ieeexplore.ieee.org/abstract/document/7160662. Acesso em: 01 nov. 2023.

MURDOCH, W. J.; SINGH, C.; KUMBIER, K.; ABBASI-ASL, R.; YU, B. Interpretable machine learning: definitions, methods, and applications. *arXiv preprint arXiv:1901.04592*, 2019. Disponível em: https://arxiv.org/abs/1901.04592. Acesso em: 02 set. 2023.

SHEN, B. W.; SETHI, G.; ZAKKA, K.; MOINDROTA, O.; DARDNER, R.; KULAL, S. *CS231n Convolutional Neural Networks for Visual Recognition*. 2020. Disponível em: https://cs231n.github.io/neural-networks-3/. Acesso em: 29 out. 2023.

SKOUDIS, E.; LISTON, T. *Counter hack reloaded: a step-by-step guide to computer attacks and effective defenses*. Prentice Hall PTR, 2005. Disponível em: https://dl.acm.org/doi/abs/10.5555/1076521. Acesso em: 12 out. 2023.

YARALLY, T.; CRUZ, L.; FEITOSA, D.; SALLOU, J.; DEURSEN, A. V. Uncovering energy-efficient practices in deep learning training: Preliminary steps towards green ai. In: IEEE. *2023 IEEE/ACM 2nd International Conference on AI Engineering–Software Engineering for AI (CAIN)*. 2023. p. 25–36. Disponível em: https://arxiv.org/pdf/2303.13972.pdf. Acesso em: 25 out 2023.