
UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Gabriel Araújo Miranda Santos

# Identificação de ameaças em redes sem fio utilizando TinyML

Campina Grande - PB

Outubro de 2024

**Gabriel Araújo Miranda Santos**

# Identificação de ameaças em redes sem fio utilizando TinyML

Monografia de Graduação apresentada ao Departamento de Engenharia Elétrica do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande como requisito parcial para a obtenção do grau de bacharel em Engenharia Elétrica.

Orientador:

Prof. Dr. Kyller Costa Gorgônio

Universidade Federal de Campina Grande - UFCG

Departamento de Engenharia Elétrica - DEE

**Campina Grande - PB**

Outubro de 2024

*Dedico este trabalho a Deus, autor da minha vida, e a família que Ele mesmo me deu. A minha esposa, a quem dedicar minha vida é pouco. A meus pais, que se dedicaram em tornar esse momento possível.*

# Agradecimentos

Agradeço primeiramente a Deus, que me deu a vida e me permitiu chegar até aqui. Ele que é a origem, o sustento e o fim de todas as coisas. Glória, pois, somente a Ele para todo sempre. Amém.

Agradeço a minha esposa, Ana Raquel, pelo apoio em todo tempo. Pelo calor nos dias frios. Por trazer clareza aos meus olhos, mesmo inundados por lágrimas. Pela personificação da persistência que ela é, e que me ensinou todos os dias com a sua própria vida.

Agradeço aos meus pais, Antônio Marcos e Maria do Carmo, aqueles a quem devo tudo o que sou e tudo o que consegui. Se hoje estou aqui, foi porque vocês fizeram o impossível para tornar isso real. Obrigado por todo investimento. Tenho dívida eterna com vocês.

Agradeço ao professor, e meu orientador, Kyller Gorgônio, que ensinou além da sala de aula. Agradeço pela paciência e dedicação durante as disciplinas e durante este trabalho. Seu empenho e disposição foram combustível para mim.

Agradeço aos professores Wamberto e Jalberth pelos ensinamentos, ao professor Gutemberg por toda prestatividade característica da sua pessoa. Agradeço ao professor Angelo Perkusich, com quem muito aprendi e que contribuiu neste trabalho.

Agradeço aos companheiros do Programa de Educação Tutorial, o PET Elétrica. Tempo precioso pude passar com eles, momentos de aprendizado, trabalho, desenvolvimento. Obrigado por terem moldado minha carreira durante a graduação.

Agradeço aos meus amigos e referências acadêmicas Arthur Dimitri e Humberto de Brito, que por seu exemplo me ensinam a ser melhor. Ter como amigos aqueles em quem me inspiro é uma honra. Também agradeço a Antonio Lúcio, quem segurou minha mão e me ensinou com diligência e empatia sobre ciência de dados. Ter esse tema em meu TCC só foi possível pelo que com ele aprendi.

Agradeço aos amigos de jornada que tornaram esse trajeto proveitoso. Obrigado a, Ana Flávia, Alexsandra Souto, Biancca Cavalcante, Jamilson Santos, Julia Ramalho, Roberto Dourado e Samara Cardoso. A companhia de vocês foi um privilégio.

Agradeço ao grupo Missão Federal que teve grande participação em toda minha formação. Ter irmãos por perto com quem podia aprender para a vida não tem preço.

Por fim, agradeço a todos quantos contribuíram com minha formação. Todos que, por algum momento, ajudaram e tornaram possível esse momento.

*"E os homens vão para o exterior para admirar as alturas das montanhas, as ondas poderosas do mar, as marés largas dos rios, a bússola do oceano e os circuitos das estrelas, mas passam por cima do mistério de si mesmos sem pensar."*

***Agostinho de Hipona***

# Resumo

Neste trabalho é proposto o desenvolvimento de um modelo de aprendizado de máquina para a identificação de ataques em redes sem fio, utilizando dados de tráfego da base UNSW-NB15. Serão treinados, testados e validados três modelos distintos: *Random Forests* e Redes Neurais Convolucionais. O objetivo é desenvolver um modelo que possa ser adaptado para a plataforma TinyML, permitindo sua implementação em dispositivos com recursos computacionais limitados.

**Palavras-chave:** Aprendizado de máquina, Redes Neurais Convolucionais, TinyML, UNSW-NB15.

# Abstract

This research proposes developing a machine learning model for identifying attacks in wireless networks using traffic data from the UNSW-NB15 dataset. Three distinct models will be trained, tested, and validated: Random Forests and Convolutional Neural Networks. The goal is to develop a model that can be adapted for the TinyML platform, enabling its implementation on devices with limited computational resources.

**Keywords:** Machine Learning, Convolutional Neural Networks, TinyML, UNSW-NB15.

# Lista de Figuras

# Lista de Tabelas

# Lista de abreviações

CNN - Convolutional Neural Networks

CPU - Central Processing Unit

CSV - Comma Separated Values

DoS - Denial of Service

FP - False Negative

FTP - File Transfer Protocol

GB - Gigabytes

GPU - Graphics Processing Unit

HTTP - Hypertext Transfer Protocol

HTTPS - Hypertext Transfer Protocol Secure

IA - Inteligência Artificial

ICMP - Internet Control Message Protocol

IP - Internet Protocol

ISP - Internet Service Provider

MAE - Mean Absolute Error

ML - Machine Learning

NaN - Not a Number

RAM - Random Access Memory

ReLU - Rectified Linear Unit

SMTP - Simple Mail Transfer Protocol

SSNMP - Simple Network Management Protocol

TCP - Transmission Control Protocol

TFLite - TensorFlow Lite

TinyML - Tiny Machine Learning

TN - True Negative

TP - True Positive

TPU - Tensor Processing Unit

UDP - User Datagram Protocol

UNSW - University of New South Wales

# Sumário

# Capítulo 1

# Introdução

A internet é uma rede de computadores que interconecta bilhões de dispositivos eletrônicos ao redor do mundo [1]. O autor James Kurose, citando dados de uma pesquisa feita por Gartner em 2014, diz que em 2015 existiam cerca de 5 bilhões de dispositivos conectados à Internet e que esse número chegaria a 25 bilhões em 2020. Além disso, era estimado que, em 2015, cerca de 40% da população mundial eram usuários da rede. Para 2024, segundo Statista [17], considerando uma população mundial de quase 8 bilhões, estima-se que 66% dela tem acesso à Internet.

Estes números são alarmantes, não somente pela vasta quantidade de pessoas com acesso ainda em 2015, mas também pelo crescimento desse número, crescendo 26% em menos de uma década. Em contrapartida, tem se tornado mais comum a ocorrência de ataques cibernéticos e fraudes digitais. Em 2018, 100 mil clientes do Banco Inter tiveram suas informações pessoais vazadas por causa de ataque *hacker* [18].

Iniciativas tem sido tomadas visando aumentar a segurança desses ambientes virtuais. A cibersegurança tem sido tópico relevante nos dias atuais diante de uma sociedade cada vez mais usufruidora das facilidades que a Internet oferece.

Os dias atuais também tem sido marcados com a democratização da Inteligência Artificial (IA), estando presente em muitos contextos e aplicações. Esta tecnologia já é presente em assistentes virtuais para *smartphones*, redes sociais, mecanismos de pesquisa, atendimento virtual, entre outros exemplos. Assim, a IA tem se mostrado uma útil ferramenta no desempenho de diversas tarefas.

Diante do cenário delicado em questões de segurança e a flexibilidade de uso da Inteligência Artificial, foi investigado a respeito da união entre essas duas esferas. Usando

artigos publicados sobre a utilização de Redes Neurais para detecção de ataques a partir de dados lidos em uma rede como ponto de partida, foi definido o desenvolvimento de um modelo de rede neural artificial que fosse capaz de identificar ataques, mas que também fosse compatível para dispositivos cujo poder computacional é limitado.

Para tanto, o modelo a ser desenvolvido precisa se apresentar confiável em suas inferências e também adaptável a um ambiente restrito de recursos, como um sistema embarcado. A fim de contemplar essas necessidades, o modelo desenvolvido se baseia em uma rede neural convolucional desenvolvida utilizando o TensorFlow, que possui recursos para conversão de modelos para TinyML.

## 1.1 Objetivo Geral

Desenvolver um modelo de inteligência artificial capaz de identificar ataques em redes sem fio compatível com ambientes de TinyML.

## 1.2 Objetivos específicos

- Comparar diferentes modelos para identificar qual atende melhor a necessidade;
- Identificar padrões de anomalias no acesso à rede para rotulação de situação maliciosa;
- Validar o modelo utilizado por meio de uma confiável taxa de acertos nos testes.

## 1.3 Estrutura do trabalho

Este trabalho foi dividido em 5 capítulos. O presente capítulo (**Capítulo 1**) introduz a temática que norteia o trabalho, bem como apresenta os objetivos que esse trabalho possui alcançar.

No **Capítulo 2** é apresentada a teoria na qual se baseia todo o desenvolvimento do trabalho. Aborda sobre o tópico de redes de computadores, os conceitos da Inteligência Artificial, a segurança cibernética e, por fim, as métricas usadas para avaliar modelos.

No **Capítulo 3** é apresentado todo o processo de desenvolvimento do trabalho. Aborda a definição dos dados a serem usados, a preparação deles, definição de modelos e

o treinamento de cada um.

No capítulo seguinte, o **Capítulo 4**, estão os resultados obtidos com os modelos apresentados no capítulo anterior e discussões a respeito deles.

No **Capítulo 5**, estão as conclusões deste trabalho, discutindo acerca das dificuldades e a perspectiva futura da continuidade das atividades.

# Capítulo 2

# Fundamentação Teórica

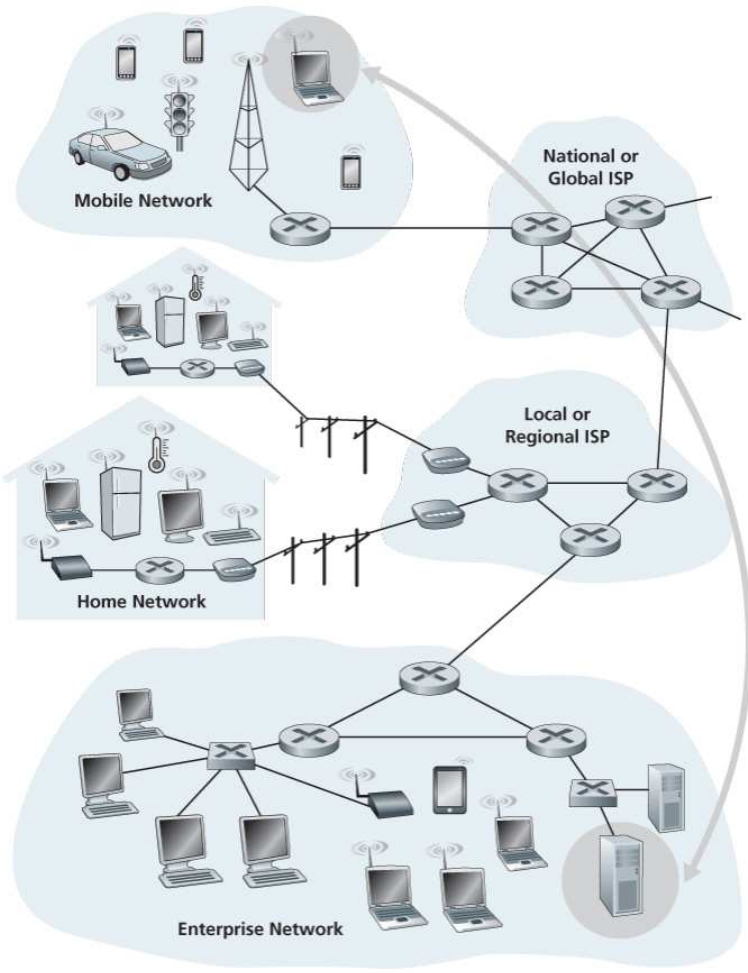
Neste capítulo serão abordados os tópicos basilares para o desenvolvimento deste trabalho. Inicialmente, serão apresentados os principais conceitos sobre redes de computadores, abordando os protocolos de comunicação e a monitoração das redes. A segurança é abordada em seguida, sendo esse um fator relevante para o estudo. Em seguida, será discorrido acerca da inteligência artificial, restringindo o escopo ao ramo de *machine learning*, apresentando as redes neurais, o TinyML e uma breve exposição do relacionamento entre IA e cibersegurança. Como terceiro tópico, métricas usadas na avaliação de modelos serão apresentadas.

## 2.1 Redes de Computadores

O termo redes de computadores refere-se a um conjunto interconectado, por meio de elementos físicos ou não, de dispositivos como *smartphones*, relógios, automóveis, sistemas de segurança doméstica, a título de exemplo, estes são denominados *hosts*. Esses dispositivos são conectados entre si por meio de links de comunicação e comutadores de pacotes e todos são conectados a internet, a maior rede de computadores atualmente, por meio de servidores de serviço de internet (ISPs, do inglês *Internet Service Providers*).

A Figura 2.1 apresenta uma ilustração de uma topologia de rede, mostrando os dispositivos *hosts* como destinos finais da linha de comunicação, os ISPs, sendo aplicados em diferentes setores de aplicação, mostrando a variedade de dispositivos que podem compor uma rede.

Figura 2.1: Topologia dos componentes de uma rede de internet




Fonte: KUROSE, ROSS, 2017

### 2.1.1 Conceitos fundamentais

Para que uma rede tenha acesso à Internet, os *hosts* devem ser conectados a um **ISP**, o qual provê conexões em diferentes setores de atuação, seja residencial, pública, industrial, entre outras. A comunicação entre os dispositivos de uma rede, o ato de enviar e receber mensagens, acontece por meio de **protocolos**. "Um protocolo", (KUROSE, 2016 [1]), "define o formato e a ordem das trocas de mensagens entre dois ou mais entidades de comunicação, bem como as ações tomadas na transmissão e/ou recepção de uma mensagem ou outro evento", ou seja, é um padrão estabelecido para que a mensagem enviada pelo emissor seja adequadamente recebida e entendida pelo receptor.

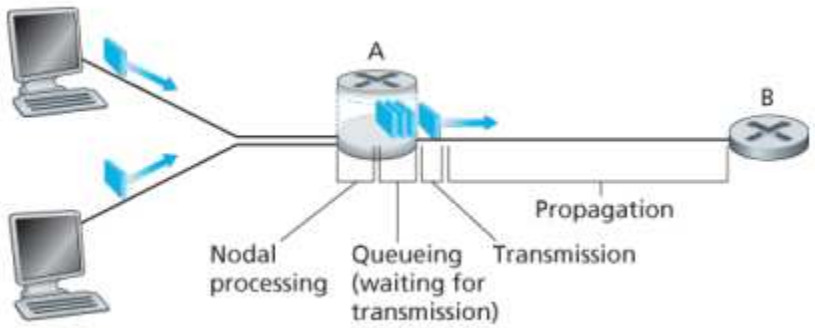
De acordo com o protocolo estabelecido, o conteúdo da mensagem é segmentado e são adicionados bytes de cabeçalhos para cada segmento, o resultado do conjunto de

informações segmentadas é chamado de **pacotes**, e estes são enviados através da rede pelos links de comunicação e o percurso entre o emissor e o receptor é chamado de **rota** ou **caminho**. O **TCP** (*Transmission Control Protocol)* e o **IP** (*Internet Protocol*) são os dois mais importantes protocolos na Internet. Os principais protocolos de Internet são conhecidos como **TCP/IP**, um conjunto de protocolos que combina o TCP e o IP.

Na etapa de transmissão de informação, cada comutador de pacote possui um ***buffer* de saída** que armazena o dado que está para ser enviado. Esse elemento é importante no caso em que o emissor que possui uma informação a ser transmitida encontra o *link* de comunicação ocupado, nessa situação, a informação é armazenada no *buffer* até que possa ser enviada. Contudo, quando o *buffer* se encontra totalmente cheio, a informação que está pra ser enviada não possui lugar para ser armazenada e, assim, ocorre **perda de pacote**.

Em um cenário ideal, as perdas são completamente evitadas e nenhuma informação é comprometida. Todavia, sistemas físicos reais estão sujeitos a interferências que comumente ocasionam perdas ou **atrasos** que devem ser considerados. Em cada nó presente em uma rota ocorrem atrasos que, no fim de todo o trajeto, ocasionam um maior atraso na transmissão da mensagem. As perdas, por sua vez, acontecem devido as limitações de capacidade das filas ou *buffers*, quando estão lotadas o roteador remove o pacote excedente e este é perdido.

Figura 2.2: Ilustração de atraso em um *buffer* de um nó




Fonte: KUROSE, ROSS, 2017

Todos os componentes de uma rede possuem uma organização sequencial e estão em uma arquitetura por camadas que orquestra todos os processos, facilitando o entendimento do papel que cada um desses elementos desempenham no sistema.

### 2.1.2 Arquitetura TCP/IP

A arquitetura baseada em **camadas** organiza a rede em seções que executam diferentes ações, ou **serviços**, mas que são essenciais para o fluxo de informações no sistema. Cada camada possui um protocolo próprio em sua implementação de acordo com o tipo de ação que ela opera. As camadas são:

- camada de aplicação;
- camada de transporte;
- camada de rede;
- camada física.

A **camada de aplicação** está no topo da escala de abstração, sendo a camada de mais alto nível. Nela estão hospedadas as aplicações da rede e os protocolos que são comumente usados são o HTTP (*Hypertext Transfer Protocol*), SMTP (*Simple Mail Transfer Protocol*) e FTP (*File Transfer Protocol*), que serão aprofundados na seção seguinte.

A **camada de transporte** transfere mensagens entre a camada de aplicação e destinos finais da aplicação. Os pacotes portadores das mensagens nessa camada são chamados de **segmentos**. Nessa camada, os protocolos que a compõem são o TCP e o UDP (*User Datagram Protocol*). Os serviços do TCP incluem a garantia de entrega da mensagem da camada de aplicação ao destinatário, além de garantir que emissor e receptor estão em sintonia, um exemplo da sua aplicação é o serviço Web que preza pela segurança dos dados (HTTP/HTTPS). O protocolo UDP, por sua vez, não possui essa garantia, o que o torna mais leve, até mesmo eficiente, e adequado para certos tipos de transmissão cujos dados são transferidos em tempo real, como por exemplo videoconferências ou sistemas de transmissão de áudio.

A **camada de rede** é responsável por mover os pacotes, agora chamados **datagramas**, de um *host* a outro por meio de uma camada de *link* responsável pelo encapsulamento dos pacotes. O protocolo da camada de transporte em um *host* endereça o segmento e o envia para a camada de rede. A camada de rede, por sua vez provê a entrega desse segmento para a camada de transporte no *host* de destino. O protocolo

que está nessa camada é o IP (*Internet Protocol*), o qual define os campos no datagrama bem como como os sistemas atuam neles. Todos os componentes de uma rede de Internet devem ter, em sua camada de rede, o protocolo IP, pois ele é o protocolo que une todos os elementos que fazem parte da internet.

O papel da **camada física** é mover *bits* de um *frame* de um nó a outro e os meios pelo qual esse movimento é feito depende do material pelo qual a transmissão será feita, nessa seção como exemplo tem-se os fios de cobre, fibras óticas, entre outros.

### 2.1.3 Protocolos de comunicação

Alguns dos principais protocolos de comunicação serão apresentados nesta seção, sendo eles pertencentes à camada de transporte, de rede ou de aplicação.

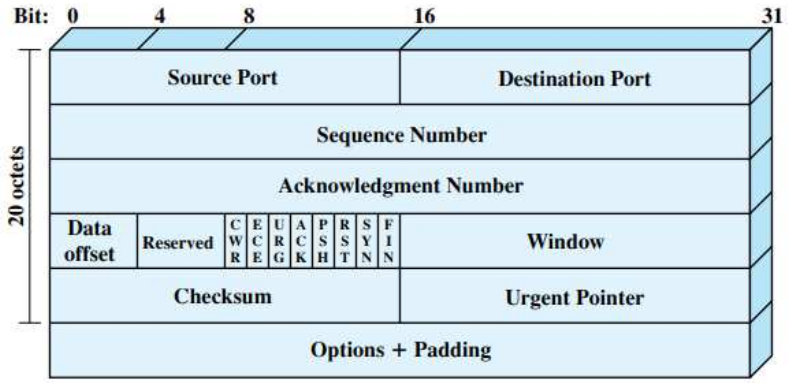
#### TCP

O protocolo de controle de transmissão (TCP) é assim definido pela sua característica de prover uma comunicação confiável entre pares ao longo de uma variedade de redes de internet, confiáveis ou não [3]. O TCP, ordinariamente, define quando são acumulados dados suficientes para formar um segmento para transmissão e, no lado do receptor, garante com que os dados sejam entregues da mesma maneira. Além disso, é papel do protocolo informar ao TCP destinatário se o dado no fluxo de transmissão é urgente, para que o usuário no destino tome uma ação apropriada.

O TCP utiliza um único tipo de unidade de dados de protocolo, o segmento (como apresentado na seção 2.1.2). Em seu cabeçalho, ou **header**, estão contidas todas as informações necessárias para os mecanismos do protocolo. Os campos que o compõem são:

- porta de origem, 16 bits;
- porta de destino, 16 bits;
- número de sequência, 32 bits;
- número de reconhecimento, 32 bits. Esse campo contém a sequência de números dos próximos dados que o TCP espera receber;
- *data offset*, 4 bits. Número de palavras de 32 bits no cabeçalho;

Figura 2.3: Estrutura de um cabeçalho TCP




Fonte: STALLINGS, 2006

- reservado, 4 bits;
- *flags*, 6 bits. As *flags* são: CWR, ECE, URG, ACK, PSH, RST, SYN e FIN;
- janela, 16 bits. Contém o número de octetos[1] de dados;
- checksum, 16 bits. O complemento de um da soma de todas as palavras de 16 bits em um segmento;
- ponteiro de urgência, 16 bits. Esse valor, quando adicionado, contém o número de sequência do último octeto de um dado urgente, o que sinaliza para o receptor quantos dados urgentes serão recebidos;
- opções, tamanho variável.

Os mecanismos do TCP podem ser visualizados em categorias: estabelecimento de conexão, transferência de dados e rescisão de conexão.

O **estabelecimento de conexão** sempre utiliza um *handshake*[2] de três vias. Para iniciar a conexão, o cliente envia um SYN com seu respectivo número de sequência e o receptor responde com um SYN e definindo a *flag* ACK, o que agora faz indica que o receptor está aguardando receber informação. O protocolo TCP suporta apenas uma conexão entre um par de portas por vez, contudo, uma porta pode possuir múltiplas conexões com outras portas.

[1] Um octeto é uma sequência de oito bits, frequentemente referida como sendo um byte

[2] O *handshake* é um processo que visa garantir a sincronia entre cliente e servidor antes da transmissão de dados iniciar

Figura 2.4: Estrutura de cabeçalho do UDP


Fonte: STALLINGS, 2006

A **transferência de dados** pode ser visualizada como um fluxo de octetos. Em cada segmento, está contido o número de sequência do primeiro octeto no campo de dados. Os dados são armazenados pela entidade de transporte tanto na transmissão como na recepção. Para o envio dos dados, a *flag* PUSH é utilizada para forçar o emissor a transmitir todos os dados acumulados. Nesse processo, o usuário pode especificar um trecho de dados como urgente, nessa situação, o receptor recebe o alerta de que dados urgentes estão sendo recebidos.

Por fim, a **rescisão de conexão** ocorre quando cada TCP (do emissor e do receptor) emitem um CLOSE[3]. A entidade de transporte, assim, atribui 1 ao bit FIN no último segmento enviado, encerrando a conexão.

### UDP

O protocolo de datagrama do usuário (UDP) também faz parte do conjunto TCP/IP, mas diferencia-se do TCP no fato de que o UDP provê serviços sem conexão para procedimentos em nível de aplicação. Por certo ponto de vista, pode-se afirmar que, portanto, o UDP é um protocolo não confiável por não garantir proteção e segurança na transmissão dos dados, contudo, existem diversas situações em que essas características são vantajosas quando comparadas a uma estrutura de grande volume e criteriosidade. Por exemplo, recebimento de dados obtidos por sensores em que a perda ocasional de dados pode não gerar preocupação pois em breve um novo dado é logo recebido.

O cabeçalho do protocolo UDP abriga poucas informações, sendo elas a porta de origem e a de destino, o comprimento do segmento, considerando o cabeçalho e o dado propriamente dito, e o Checksum que nesse contexto é opicional. Durante a transmissão, se algum erro for detectado, todo o segmento é descartado e nenhuma outra ação é tomada.

[3]O CLOSE é um tipo de serviço primitivo, o qual é um método usado que solicita ou indica eventos e é aplicado no propósito de facilitar a comunicação entre camadas

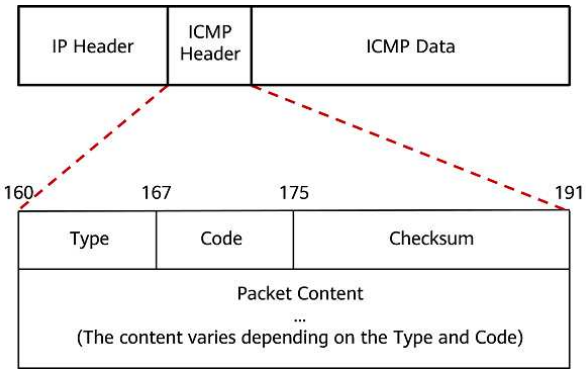
### ICMP

O protocolo de controle de mensagem da Internet (ICMP) é um protocolo da camada de rede 2.1.2 usado na comunicação entre dispositivos, principalmente atuando na identificação se um dado está chegando ou não no seu destino pretendido. A aplicação primária desse protocolo é a reportação de erros, quando pacotes transmitidos são impossíveis de enviar e o roteador remove-o do *link* de comunicação, ou ocasionados por outros fatores. Outra aplicação é a realização de diagnóstico da rede pelo fato do ICMP acessar os caminhos entre os dispositivos e poder identificar as fontes de atraso na transmissão dos pacotes.

Diferentemente do TCP ou UDP, o ICMP não está associado à camada de transporte e sim à de rede, isso o torna um protocolo não baseado em conexão, ou seja, um dispositivo não precisa abrir conexão com outro, nem usar *handshake*, para poder enviar uma mensagem, consequentemente, também não permite o endereçamento a uma porta específica em um elemento do sistema. Um cabeçalho de uma mensagem segundo padrão ICMP consiste em:

- tipo, 8 bits. O tipo da mensagem;
- código, 8 bits. Especifica os parâmetros da mensagem;
- checksum, 16 bits;
- parâmetros, 32 bits. Usado para especificar parâmetros maiores.

Figura 2.5: Estrutura de mensagem do ICMP




Fonte: FULIN, 2021)

**HTTP**

O protocolo de transferência de hipertextos (do inglês *Hypertext Transfer Protocol*) é o protocolo do WWW (*World Wide Web*) e é usado para transmissão de qualquer tipo de informação, seja de texto, hipertexto, áudio, imagens ou qualquer informação suportada pela Internet. O seu uso mais comum é entre navegadores e servidores Web. O HTTP faz uso do TCP para garantir a confiabilidade nas suas transmissões, contudo, casa uma delas é tratada independemente da outra, o que torna o protocolo *stateless* (não armazena estados), de forma prática, cada conexão é interrompida logo que a transmissão de dados encerra.

**FTP**

O FTP (*File Transfer Protocol*) é utilizado na transmissão de arquivos, seja texto ou binário, de um sistema para outro. É um protocolo baseado no TCP, de modo que para iniciar o envio de mensagens uma conexão TCP é feita e, nessa conexão, é permitido a transferência de informações sensíveis como senha e identificação de um usuário. Uma vez que a transferência de arquivo é aprovada, uma segunda conexão TCP é feita para emitir os dados e o arquivo é enviado ao longo da conexão dos dados sem a necessidade de cabeçalhos no nível de aplicação. Quando a transmissão é concluída um sinal é emitido indicando a completude da ação e que está disponível para uma nova transmissão.

### 2.1.4 Monitoração e gerenciamento de redes

Através do uso de *software*, uma rede pode ser observada a nível de pacotes, sendo visualizado a transmissão de informação e, a partir disso, diagnosticar a rede, identificar sobrecargas ou mal funcionamento. Dispositivos roteadores ou comutadores são normalmente alvos do monitoramento, tendo em vista a sensibilidade da sua função e as vulnerabilidades que podem enfrentar.

Os tipos principais de ferramentas que podem ser utilizadas para o monitoramento são:

1. **Baseadas em SNMP**: utilizam do protocolo de gerenciamento de rede simples (*Simple Network Management Protocol*). É o tipo de ferramenta mais usada e, com ela, é possível monitorar em tempo real o fluxo da rede;

2. **Baseadas em fluxo**: monitoram o tráfego da rede;

3. **Soluções de monitoramento ativo de rede**: injetam pacotes dentro da rede para obtenção dos dados.

## 2.2 Segurança

Dado o crescimento da Internet, essa rede tem se tornado crítica para muitas instituições atualmente. Além de indústrias, pessoas físicas também tem tido mais acesso a dispositivos *smarts* (com acesso à Internet) para as mais diversas atividades. Em qualquer contexto de conexão, os aspectos que são importantes e desejáveis para uma autêntica segurança em uma rede podem ser listados com base no que é apresentado em [1], são eles:

- **confidencialidade** implica que apenas o emissor e o receptor devem entender o conteúdo da mensagem transmitida. Um caminho para isso é a criptografia;

- **integridade da mensagem** remete que a mensagem não é modificada durante sua transmissão, sendo entregue o mesmo conteúdo que foi enviado;

- **autenticação ponta a ponta** visando a confirmação da identidade dos envolvidos na comunicação. Tanto o emissor como o receptor devem poder identificar que estão se comunicando com aquele a quem desejam se comunicar;

- **segurança operacional** requer que as redes devem ser seguras para serem acessadas por seus usuários e devem impedir que seja possível sua infecção por *worms*.

Com a exposição de tantas informações em uma rede, indivíduos mal intencionados encontram oportunidades para atacarem e causar males aos dispositivos e seus respectivos usuários, violando a privacidade e roubando o acesso às contas de um usuário. Diante do exposto, a segurança cibernética se torna um tópico central no campo de redes de computadores e um ponto de partida é a compreensão de quais são as vulnerabilidades de uma rede que possibilitam a ocorrência de ataques nela e como esses ataques podem ser classificados.

### 2.2.1 Classificação de ataques

Dentre os mais variados tipos de ataques existentes na atualidade, uma observação deles em duas categorias é proposta por [3].

#### Ataques passivos

Este tipo de ataque é baseado na escuta, ou monitoramento, das transmissões com o objetivo de obter as informações que estão sendo enviadas de um ponto a outro. Como não há alteração ou interferência no conteúdo das mensagens, ataques nessa categoria são mais difíceis de identificar, pois o conteúdo permanece inalterado e tanto o emissor como o receptor mantém a comunicação normalmente sem identificar que existe um terceiro que está escutando tudo o que se transmite. Como maneira de evitar esse tipo de ataque a criptografia é uma opção comumente utilizada para prevenir que o conteúdo da mensagem que pode ser obtido seja também entendido.

#### Ataques ativos

Por outro lado, ataques ativos envolvem modificações no fluxo de dados, podendo até substituir os dados enviados por outros dados sinteticamente produzidos e falsos. Algumas categorias em que esse tipo de ataque se encaixa são apresentadas na lista:

- **mascarado**, um ataque mascarado ocorre quando uma das entidades na comunicação na verdade é alguém não autorizado fingindo ser um elemento confiável da linha;
- ***replay***, envolve a obtenção dos dados e sua retransmissão para produzir um efeito não autorizado;
- **modificação de mensagens**, alteração do conteúdo que uma mensagem carrega gerando, assim, um efeito não autorizado.

Além dessas, outra categoria é chamada de ***denial of service***, a qual será apresentada a seguir.

### 2.2.2 *Denial-of-service*

Uma classe comumente utilizada para atacar redes é denominada ataques *denial-of-service* (DoS). Esse tipo de ataque rende uma rede, *host*, ou outro componente e o torna inutilizável para usuários legítimos. Estima-se que os ataques desse tipo ocorrem na casa dos milhares por ano [9], o que o torna um caso a ser atentamente observado. Os fatores que mais contribuem para a ocorrência de um ataque DoS são:

- **ataque de vulnerabilidade**, envolve o envio de mensagens para dispositivos vulneráveis, podendo ocasionar a interrupção de sua operação;
- **inundação da largura de banda**, é o envio de inúmeros pacotes a ponto de impedir que pacotes legítimos sejam recebidos pelo *host*;
- **inundação de conexão**, é a abertura de inúmeras conexões TCP ao ponto que o *host* não consegue mais se conectar com dispositivos legítimos.

Outro exemplo de ataque comum que se torna importante de ser apresentado é o ***malware***.

### 2.2.3 *Malware*

O acesso a vários conteúdos em páginas Web, aplicativos, sites de *streaming*, redes sociais, entre outros, leva o usuário a ter acesso a uma gama de informações que por muitas vezes é saudável, mas que pode, também, abrigar conteúdo malicioso comumente denominado como *malware*[4]. Uma vez infectado, diversas ações maliciosas podem ser executadas no dispositivo comprometido, como deletar arquivos ou informações sensíveis, espionar o comportamento do usuário e reportar tudo para o indivíduo que fez o ataque.

O *malware* é auto-replicante, ou seja, ao infectar um *host*, a partir deste ele pode procurar outros dispositivos para infectar e, ao encontrar, o processo se repete. As formas que o *malware* pode se reproduzir são como **vírus** ou ***worms*** (do inglês, vermes). Na primeira forma, o *malware* requer uma interação do usuário para infectar o dispositivo, como um download de arquivo. Como ***worm***, a interação já não é necessária, por exemplo,

[4]Segundo a IBM, !"[...]*malware* é qualquer código de software ou programa de computador [...], escrito intencionalmente para prejudicar os sistemas de computador ou seus usuários. Quase todos os ataques cibernéticos modernos envolvem algum tipo de *malware*"[8]

o acesso do indivíduo a uma rede vulnerável na qual um *hacker* pode injetar *malware*. Em ambas as formas, as informações do usuário são colhidas e transmitidas para quem fez o ataque.

### 2.2.4 *Sniffers*

O acesso sem fio à Internet traz consigo a facilidade de poder se conectar a partir de dispositivos móveis de qualquer lugar. Contudo, essa facilidade ocasiona brechas de segurança por acabar dando espaço para que um dispositivo receptor passivo seja inserido no meio e atue recebendo uma cópia da informação de todo pacote que transita pelo *link*. A esse recebedor passivo dá-se o nome de ***sniffer*** de pacotes[5], que também pode ser inserido em redes cabeadas.

Pelo fato de que um *sniffer* não interfere na transmissão, apenas recebe informação, ele se torna mais difícil de detectar. Esse fator passivo e de não interferência faz com que existam ferramentas gratuitas que realizam essa atividade. O perigo reside quando um *hacker* obtém acesso à rede, pois estando nela, pode obter as informações através do *sniffer*.

[5]O que em tradução literal seria "farejador de pacotes"

## 2.3 Inteligência artificial

A Inteligência Artificial (IA) ocupa posições recentes na história da ciência, especialmente na ciência de dados. Os trabalhos que envolviam esse campo de estudo iniciaram por volta da Segunda Guerra Mundial, apesar do termo só vir a ser cunhado em 1956 [5]. Atualmente, a IA é composta por vários outros campos de estudo, dentre os quais serão destacados o *Machine Learning* e as redes neurais (sendo este último pertencente ao ramo do *Deep Learning*).

Dentre uma gama de explicações sobre o que melhor se expressa com o termo "inteligência artificial", destacar-se-á a definição de Kurzweil: "A arte de criar máquinas que performam funções que requerem inteligência quando performadas por pessoas"[6]. O teste de Turing, proposto em 1950, expressa bem o critério que posteriormente foi apresentado por Kurzweil. O teste consiste em um interrogador e um interrogado. Uma máquina é aprovada no teste caso, após o interrogatório, o interrogador não ser capaz de identificar se as respostas foram postas por um computador ou um humano. Ou seja, uma máquina com habilidade de agir (responder perguntas) com uma habilidade tal qual humana.

Esse teste, apesar de antigo, continua sendo relevante nos dias atuais, fazendo pesquisadores buscarem pelo entendimento dos princípios da inteligência ao invés de simplesmente buscarem reproduzir exemplos. Para atingir tal nível de capacidade, um computador deve possuir habilidades tais como:

- **processamento de linguagem natural**, compreender e se comunicar na língua humana;
- **representação de conhecimento**, para armazenar o que aprende;
- **raciocínio automatizado**, usar informação armazenada para responder novas perguntas;
- **aprendizado de máquina**, adaptação a novas circunstâncias e capacidade de identificar padrões.

[6]Essa definição se baseia no pressuposto de que IA busca reproduzir uma máquina que aja humanamente. Além desse, outros poderiam ser assumidos como o de pensar humanamente, pensar racionalmente ou agir racionalmente. Para mais definições, ver [5].

A partir de agora, o campo será restringido ao escopo do aprendizado de máquina, do inglês, *machine learning*.

### 2.3.1 *Machine learning*

É considerado que uma máquina aprende "se ela melhora sua performance em tarefas futuras após observações acerca do mundo"[5]. A aprendizagem é uma habilidade desejável tendo em vista que os cenários nos quais as máquinas são inseridas para atuar são reais e não controlados, possuem situações que não foram mapeadas antes, mas que não isentam o computador de realizar ações quando lhe é requerido. Portanto, a máquina deve possuir capacidade tal, para que por meio de padrões possa inferir o que deve ser feito sem ter sido explicitamente programada, pois seria inviável, ou até mesmo impossível, definir rotinas a serem seguidas para cada contexto possível de ocorrer.

Denomina-se **agente** como o componente aprendiz, ou seja, a máquina. Para que um agente melhore seu aprendizado, alguns fatores devem ser considerados:

- qual componente está para melhorar;
- qual conhecimento já adquirido;
- qual representação dos dados será usada;
- qual *feedback*[7] está disponível para ser usado no processo.

Os componentes que podem ser alvo do aprendizado podem ser exemplificados como: meios de inferir propriedades a partir de uma percepção dos dados já disponíveis, predições de como o ambiente se comportará para tomada de decisões, entre outros.

O *feedback* compreende um recurso essencial para o aprendizado e a depender do modo como ele é utilizado diferentes abordagens podem ser escolhidas no processo de aprendizagem.

#### Aprendizado não supervisionado

Nessa abordagem, o agente aprende a partir da identificação de padrões nos dados fornecidos apesar de nenhum *feedback* lhe ser dado. É responsabilidade do próprio

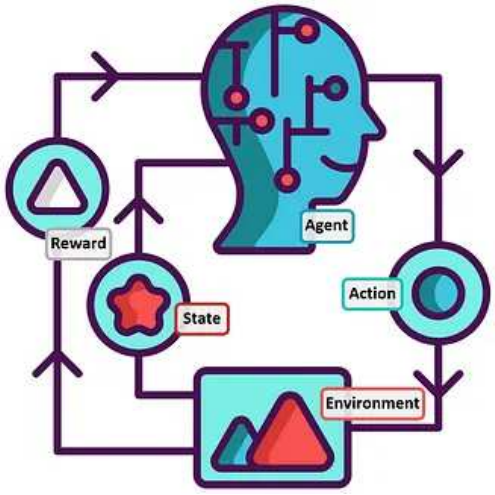
[7] *Feedback* é uma informação específica sobre a comparação entre a observação da performance ou do conhecimento de um aluno no desempenho de uma tarefa e o padrão desejado.[11]

algoritmo, na observação dos dados, identificar padrões e ser capaz de agrupar dados semelhantes. Essa tarefa de agrupamento é uma das mais comuns no uso dessa abordagem e é chamada de **clusterização** (do inglês, *clustering*).

### Aprendizado por reforço

O agente aprende a partir de *feedbacks* que podem ser positivos (recompensas) ou negativos, a depender da ação tomada por ele ser alinhada ou não com o seu propósito. Esse tipo de aprendizado pode ser ilustrado como a maneira com a qual um cachorro aprende: sistema de recompensas. No intuito de ensinar o animal a sentar ao ser solicitado a isso, caso ele obedeça recebe uma recompensa que na sua mente o fará atribuir aquela ação à uma resposta positiva do dono, caso contrário recebe uma punição que o fará associar a sua ação como algo que não deve ser feito. O mesmo raciocínio pode ser aplicado substituindo o cachorro por um computador e o dono pelo ambiente no qual o computador está. A figura abaixo ilustra essa dinâmica, um agente executa ações em um ambiente, do qual são obtidas respostas, como o estado que o sistema se encontra, a depender da ação tomada.

Figura 2.6: Ilustração do processo de aprendizado por reforço




Fonte: Disponível em `https://towardsdatascience.com/reinforcement-learning-fda8ff535bb6`

### Aprendizado supervisionado

Nessa abordagem, o agente recebe dados rotulados com os pares de entrada-saída como exemplos a serem observados para aprendizagem que acontece na etapa de treinamento. A partir da observação dessas informações, o algoritmo desenvolve uma função

que mapeia qual deve ser a saída a partir de uma entrada não observada na etapa de treino.

**Aprendizado semi-supervisionado**

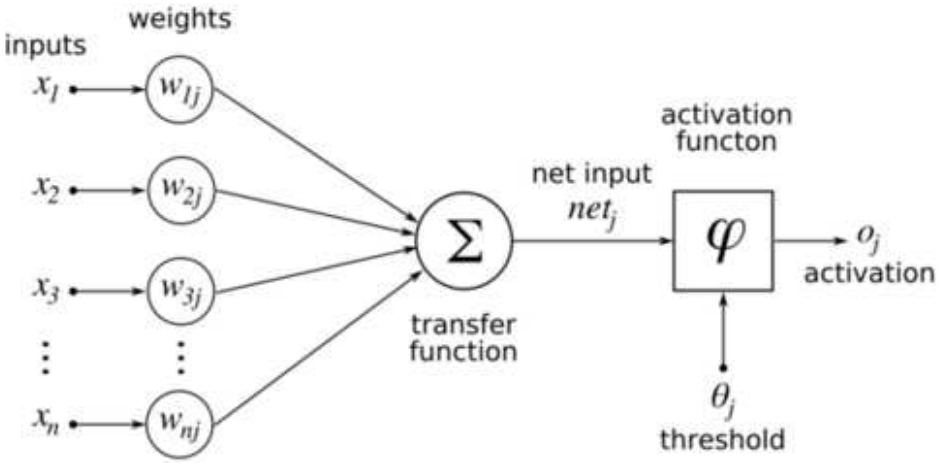
Essa abordagem é uma mesclagem entre o aprendizado supervisionado e o não supervisionado. Nela, são tanto fornecidos dados rotulados como também não rotulados. Essa técnica pode ser utilizada visando o aprendizado de funções que mapeiam saídas a partir dos dados de entrada, como também fazem o algoritmo identificar padrões nos dados e, assim, otimizar as inferências que realiza.

### 2.3.2 Redes neurais

Dentro do subconjunto do aprendizado de máquina existe ainda o *deep learning* (aprendizagem profunda) que busca desenvolver a aprendizagem com modelos que reproduzem o comportamento do aprendizado humano, conhecidos como **redes neurais artificiais** (*artificial neural networks*). A concepção do *deep learning* foi iniciada por volta da década de 50 com o Perceptron, seguida dos algoritmos de retro propagação [4]. Atualmente, diversos fatores têm contribuído para a evolução desse campo de estudo, como o aumento do poder computacional, a abundância de informações e a otimização de algoritmos. Diante disso, o uso de *deep learning* em diferentes contextos tem se popularizado, se tornando uma opção comum diante de situações que exigem aprendizado de máquina.

Por conta disso, as aplicações do *deep learning* são inúmeras, assim como o ser humano é adaptativo a diferentes segmentos por suas habilidades cognitivas, assim ocorre com modelos de aprendizagem dessa categoria pela utilização das **redes neurais**. Uma rede neural é a base do aprendizado profundo, tendo seu funcionamento sendo uma tentativa de reproduzir o comportamento do neurônio humano. A unidade básica de sua estrutura é chamada **perceptron**, os perceptrons são organizados em camadas e cada camada é ligada uma com a outra dando forma a uma rede. As conexões dentro dessa rede são definidas por pesos e parâmetros que determinam o quão relevante um dado processado é e como ele é processado desde a entrada até a saída da rede.

Figura 2.7: Estrutura de um perceptron




Fonte: REDDI, 2024

### Perceptron

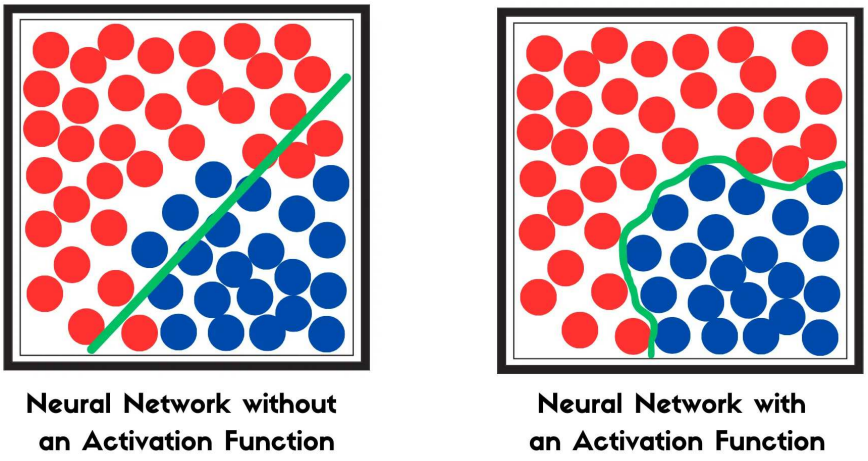
A unidade básica de uma rede complexa funciona a partir de múltiplas entradas, cada uma delas sendo a representação de uma *feature*[8]. Essas entradas são chamadas $x_1, x_2, ..., x_n$ e são acompanhadas por seus respectivos pesos $w_1, w_2, ..., w_n$. Essa estrutura é apresentada na figura 2.7, na qual estão as entradas e seus respectivos pesos sendo transmitidos a uma função de transferência e a uma função de ativação. A saída intermediária desse perceptron é definida como a soma dos produtos das entradas e seus pesos somado a um fator chamado de viés (*bias*) $b$ para uma melhor adequação da função ao conjunto de dados. A equação do perceptron pode ser definida como:

$$z = \Sigma(x_i \cdot w_{ij}) + b \tag{2.1}$$

Um perceptron pode ser configurado para tarefas tanto de regressão como de classificação. Contudo, essa forma apenas é apropriada a relacionamentos lineares entre as entradas e as saídas. Para lidar com situações em que os padrões são não-lineares, como ilustrado na figura 2.8, é utilizada a chamada **função de ativação** que é aplicada à saída linear da rede neural.

[8] *Feature*, ou atributo, é uma característica que diz respeito ao dado em questão

Figura 2.8: Ilustração de dados com relacionamento não linear

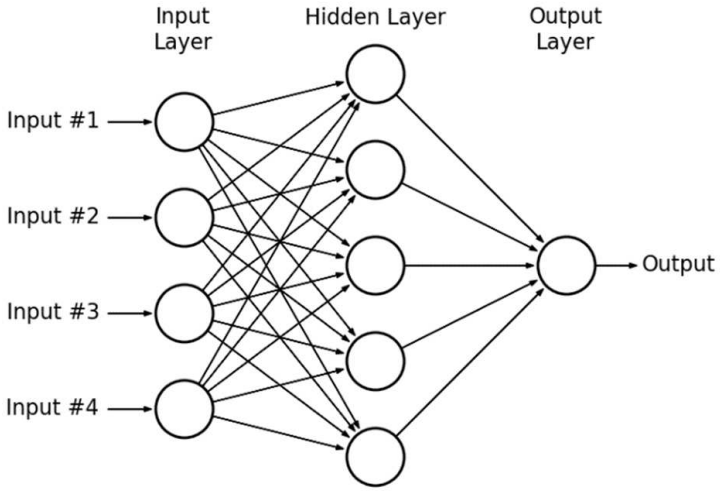



Fonte: REDDI, 2024

### Perceptrons multicamada

A formação de uma rede neural não é composta de apenas um perceptron, mas de um conjunto deles. Os perceptrons multicamada são uma evolução do modelo de perceptron, no qual camadas são interconectadas de uma maneira progressiva (*feedforward*). Cada camada consiste de numerosos perceptrons, os quais permitem a rede neural identificar relacionamentos não lineares dos dados. Uma ilustração de uma rede de perceptrons multicamada é apresentada a seguir.

Figura 2.9: Ilustração de uma rede neural multicamada composta por perceptrons




Fonte: REDDI, 2024

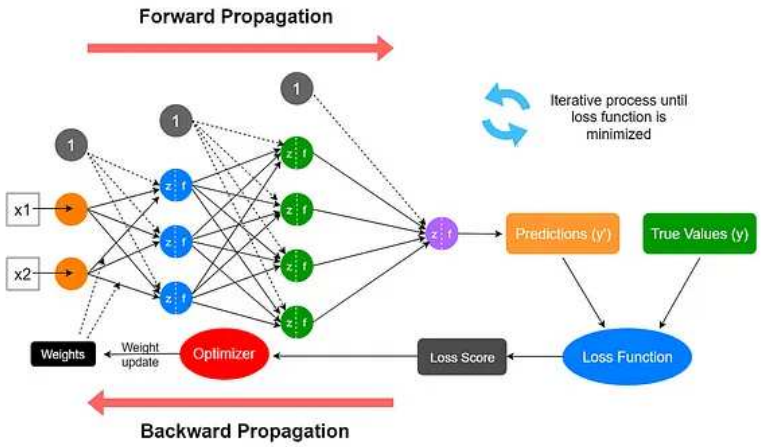
### Processo de treinamento

O fluxo de uma rede neural se baseia em receber uma entrada, realizar cálculos a partir dela e produzir uma predição baseada no que foi calculado pelos perceptrons levando em consideração os pesos de cada *feature*. Não havendo controle sobre as informações de entrada da rede, o objetivo durante o treinamento de uma RN é o ajuste dos pesos de tal forma que a predição seja a mais precisa possível.

O processo de treinamento envolve etapas chave, sendo elas a **propagação *forward*** e a **propagação *backward*** que são apresentadas na figura 2.10. Esse processo é contínuo e repetitivo, envolvendo as duas etapas em um ciclo após o outro até o momento em que a diferença entre a predição e o real valor seja ínfima, ou que o número de iterações estipulado seja atingido.

Na propagação *forward*, os dados seguem o fluxo partindo da entrada da rede até a saída. Nesse processo, cálculos com pesos randômicos são feitos na tentativa de atingir um valor razoável para uma predição. Cada camada define seus pesos ($w$) e vieses ($b$), ao final dos cálculos, os resultados são passados para a camada seguinte que irá seguir o mesmo processo até chegar na camada final cujo resultado será a predição. Essa predição é comparada ao valor esperado (valor esse que é conhecido por ser a fase de treinamento) para calcular a perda, ou a diferença entre eles. A informação de perda é um fator quantitativo para medir o ajuste dos pesos da próxima iteração.

Figura 2.10: Propagação *forward* e *backward*




Fonte: Disponível em: `https://medium.com`

Após a etapa de propagação *forward* concluir e o cálculo da perda ter sido feito,

essa informação é usada para o reajuste dos pesos a serem usados em cada camada de perceptrons. Dá-se início a etapa da propagação *backward*, na qual é determinado o gradiente de cada peso. Os gradientes descrevem a direção e magnitude na qual os pesos devem ser ajustados para que na próxima iteração a perda seja menor, indicando que o modelo está realizando predições precisas.

#### Redes Neurais Convolucionais (CNNs)

Dentre as arquiteturas possíveis para modelos de Redes Neurais a rede neural convolucional é principalmente usada para tarefas de reconhecimento de imagem ou vídeo, mas não se restringe a apenas esse segmento. Essa arquitetura é constituída de duas partes principais, a base convolucional e as camadas inteiramente conectadas.As *features* extraídas da primeira parte passa para a segunda, na qual é realizada a classificação. Esse tipo de arquitetura pode ser otimizado para sistemas embarcados usando tecnologias como quantização ou *pruning*[9] para redução do uso de memória e poder computacional.

### 2.3.3 TinyML

TinyML surge como uma opção para poder executar modelos de aprendizagem de máquina em sistemas bastante limitados em memória, armazenamento e poder computacional. Isso significa que o dispositivo responsável por gerar dados, um sistema embarcado que é equipado de sensores, é capaz de nele mesmo realizar tomadas de decisões baseadas em predições de um algoritmo nele hospedado, tornando possível ações em tempo real mesmo em ambientes em que a conectividade com um servidor robusto é inviável.

Os sistemas limitados de recursos são atendidos na medida que soluções otimizadas dentro das limitações são desenvolvidas nos algoritmos de aprendizagem, fazendo com que os modelos possam ter uma boa performance enquanto consumem poucos recursos do dispositivo. Alguns benefícios quanto ao uso do TinyML podem ser listados:

- **baixa latência** pelo fato de que a computação ocorre diretamente no dispositivo que obtém os dados, o tempo para transmissão dos dados até servidores para o processamento deles é eliminado;

[9]Do inglês significa podagem. É uma técnica usada para redução de dimensionalidade do conjunto de dados

- **alta segurança dos dados**, ainda considerando o fator de que o processamento dos dados se dá no próprio dispositivo embarcado, isso também impede que terceiros interceptem dados no meio da transmissão;

- **eficiência energética** por conta das otimizações nos algoritmos de aprendizagem, TinyML garante que pequenos dispositivos podem executar tarefas complexas sem que a bateria que o mantém ativo seja comprometida, fazendo-o operável por longo tempo.

Entretanto, o ramo do TinyML também possui desafios a serem enfrentados no seu desenvolvimento:

- **capacidades computacionais limitadas**, alguns algoritmos não podem ser convertidos para modelos de TinyML diante das restrições de *hardware*;

- **Ciclo complexo de desenvolvimento**: por conta da necessidade de otimização pelo fato de que modelos de aprendizado de máquina são complexos, o desenvolvimento de modelos para TinyML envolvem muito conhecimento de princípios de ML tanto como de sistemas embarcados;

- **compressão e otimização do modelo**, a conversão propriamente dita de um modelo de *machine learning* para um ambiente de recursos limitados como microcontroladores é um desafio pois atingir a mesma eficiência abrindo mão de poder computacional não é trivial.

### 2.3.4 IA e cibersegurança

Diante das diversas aplicações com as quais a inteligência artificial e as redes neurais podem lidar, a segurança das redes de computadores se torna mais um exemplo de contexto passível de receber os benefícios da utilização dessas tecnologias. Um sistema de cibersegurança inteligente é possível quando, ao aplicar vários métodos do campo da IA, um modelo capaz de tomar decisão em serviços na rede visando a segurança dela [12] é desenvolvido.

## 2.4 Treinamento de modelos de aprendizagem

Modelos de aprendizagem são passíveis a erros e equívocos que muitas vezes levam a predições distantes daquilo que é esperado. Por conta disso a etapa de treinamento de um modelo acontece repetidas vezes visando que a performance do modelo seja a mais precisa e confiável possível. Para que esse objetivo seja alcançado, várias técnicas são aplicadas no treinamento de uma rede neural. Também, foram desenvolvidas métricas que possibilitam a quantificação da eficiência de um modelo a partir dos seus resultados, servindo de base para observar quais nuances merecem mais atenção para ajustes.

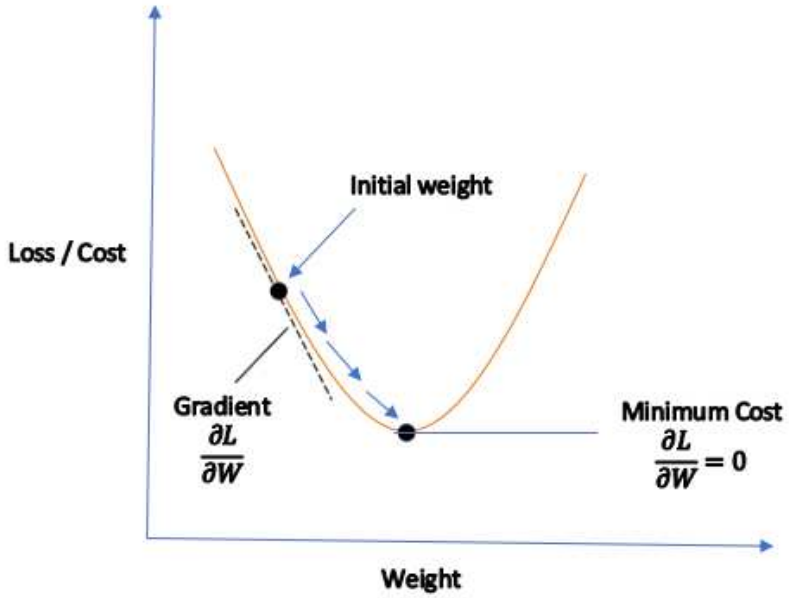
### 2.4.1 Otimização

A otimização é a técnica que busca ajustar os pesos do modelo visando melhorar o resultado de suas predições. Tendo acesso a informação do quão próximo do ideal o modelo chegou, medindo isso pela **função de perda**, esse dado é transmitido na propagação *backward* e utilizado para recalcular os novos pesos de cada camada de perceptrons.

#### Gradiente Descendente

O gradiente descendente é uma poderosa ferramenta utilizada com o propósito de recalcular os pesos visando a otimização de um modelo. Matematicamente, é tomada a derivada da função de perda, que foi definida pela rede neural, para cada peso associado a uma entrada fazendo os ajustes deles na direção que o gradiente aponta. Esse processo é repetido até que a perda seja zero, ou mínima, indicando um acerto máximo do modelo. A figura 2.11 ilustra esse procedimento de maneira gráfica, apresentando um valor inicial para o peso e o sentido descendente no resultado da derivada parcial a função de perda em relação ao peso até que este seja 0, alcançando o valor ideal para o peso.

Figura 2.11: Gradiente descendente




Fonte: Disponível em `https://towardsdatascience.com/`

### 2.4.2 Regularização

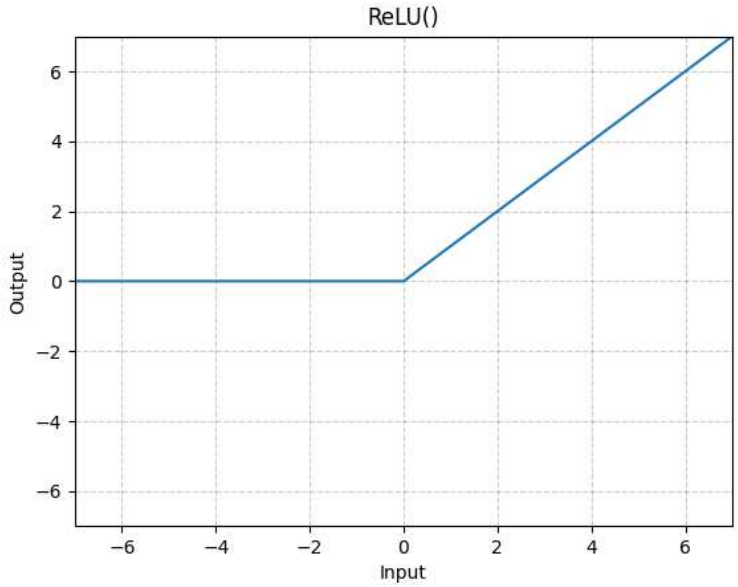
A regularização busca melhorar a performance e generalidade. Na prática, consiste em penalizar o modelo e restringi-lo buscando evitar com que ele fique muito enviesado com os dados de treinamento a ponto de não conseguir acertar nada com dados fora desse escopo (a esse acontecimento dá-se o nome de ***overfitting***).

Um método amplamente utilizado quando treinando modelos de rede neural é o *dropout*, que funciona por meio de, durante o treino, aleatoriamente leva uma fração de nós a zero, o fator que define essa aleatoriedade é definido pelo desenvolvedor e é chamado de $p$. Valores comuns de $p$ variam entre 0,2 e 0,5. No momento das predições, toda a rede neural é utilizada, mantendo todos os nós da rede.

As **funções de ativação** são essenciais para tornar a rede neural capaz de trabalhar em contextos de não linearidade. Elas devem ser diferenciáveis, ou seja, ter a derivada de primeira ordem bem definida para permitir a retro propagação e o gradiente descendente. Além disso, também devem restringir o valor de saída a um escopo que evite predições exorbitantes. Dentre elas destacam-se a ReLU e a Softmax.

A função *Rectified Linear Unit* (unidade linear retificada), ou **ReLU**, apresenta uma restrição de comportamento simples que de forma matemática pode ser expresso

Figura 2.12: Ilustração gráfica da ReLU




Fonte: Disponível em `pytorch.org`

como:

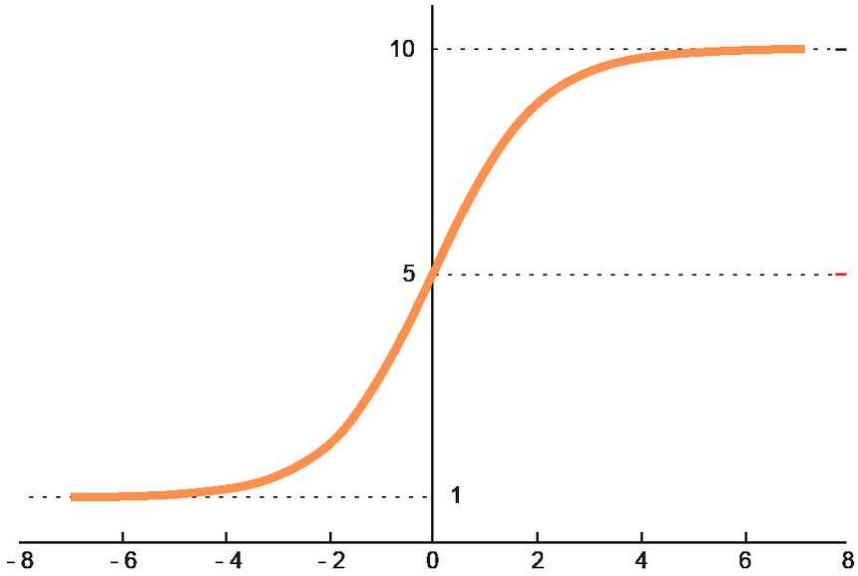
$$ReLU(x) = max(0, x) \tag{2.2}$$

O que faz com que o valor positivo da saída seja ele mesmo, mas restrinja-o para que não hajam valores negativos, atribuindo zero para cada um desse tipo que possa aparecer, como apresentado na figura 2.12.

A função **Softmax**, cujo comportamento gráfico é ilustrado na figura 2.13 (valores arbitrários), é comumente usada na última camada em tarefas de classificação, em situações em que é desejável prever a partir da probabilidade dentre as classes que um dado de entrada pode assumir. Ela é definida como:

$$\sigma(z_i)\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{para} i = 1, 2, ..., K \tag{2.3}$$

Figura 2.13: Ilustração gráfica da Softmax




Fonte: Disponível em `researchgate.net`

### 2.4.3 Métricas de avaliação

A avaliação de quão assertivo um modelo está é imprescindível para os ajustes que o levam a melhorar. As métricas de avaliação fornecem informações quantitativas relevantes de como o algoritmo está se saindo. Dentre as métricas, aqui destacam-se:

#### Acurácia

É a porcentagem de classificações corretas dentre o número total de amostras que foram avaliadas. É uma métrica simples mas que se limita a conjuntos de dados que estejam bem balanceados.

$$Acurácia = \frac{TP + TN}{Total} \tag{2.4}$$

Sendo, $TP$ os classificados corretamente como sendo uma determinada classe (*True positives*) e $TN$ os corretamente classificados como não sendo uma determinada classe (*True negatives*).

#### Precisão

Se baseia na proporção de quantas predições classificadas positivamente estão realmente certas, expressando a tendência de um modelo em classificar uma classe e o quão

susceptível pode ser a alarmes falsos.

$$Precisão = \frac{TP}{TP + FP} \tag{2.5}$$

Sendo $FP$ os classificados erroneamente como sendo de uma determinada classe (*False positives*).

***Recall***

Verifica a proporção de classificações corretas dentre todas que deveriam ter sido classificadas da mesma forma. Se as classes são apenas 0 ou 1, é a proporção entre o número de acertos em que a classe é 0, sobre o total número de casos em que a classe é de fato 0. Essa métrica avalia quão bem um modelo se sai na classificação de cada caso possível.

$$Recall = \frac{TP}{TP + FN} \tag{2.6}$$

Sendo $FN$ os classificados erroneamente como de uma classe quando na verdade são de outra.

***F1-Score***

É a média harmônica entre a precisão e o *recall*. Representa, em uma métrica, o quão bom um modelo é tanto em classificar corretamente quanto em atribuir corretamente a uma determinada classe.

$$F1 - Score = \frac{2 \cdot (\text{Precisão} \cdot \text{Recall})}{\text{Precisão} + \text{Recall}} \tag{2.7}$$

***Mean Absolute Error***

Verifica a média do módulo da diferença entre os valores reais e os valores preditos pelo modelo. Essa métrica não eleva a diferença ao quadrado (como no caso do *Mean Squared Error*, apenas considera toda diferença como de mesmo módulo, independente de para que direção o erro está. Quanto menor o MAE, indica que maior é a taxa de acerto do modelo, pois o erro é mínimo.

$$MAE = \frac{\Sigma_{i=1}^{n}|y_i - x_i|}{n} \tag{2.8}$$

Sendo, $n$ o número de dados observados, $y_i$ o valor real de cada variável predita e $x_i$ o valor predito pelo modelo.

# Capítulo 3

# Metodologia

Este capítulo apresenta o fluxo do desenvolvimento do trabalho. Nele serão apresentadas as etapas que abrangem desde a solução proposta inicialmente, a definição do conjunto de dados, a concepção dos modelos a serem utilizados, o tratamento desses dados para se adequarem às necessidades e restrições do processo de aprendizado de máquina, bem como seus respectivos treinamentos.

## 3.1 Solução Proposta

A segurança nas redes de computadores é um fator de grande importância em um contexto de alta exposição de dados na Internet. Além disso, a utilização de inteligência artificial em muitos casos tem se mostrado como eficiente para resolução de problemas. Considerando a necessidade de sistemas inteligentes capazes de identificar ataques em redes cibernéticas a partir do monitoramento dos dados que estão sendo transmitidos, este trabalho propõe um modelo de rede neural cujo objetivo é, a partir de informações de tráfego em uma rede qualquer, classificar, de maneira binária, quando um acesso se apresenta como suspeito ou não.

Para tanto, é proposto o uso de um conjunto de dados amplamente utilizado na literatura, o UNSW-NB15, criado pela Universidade de Sydney em South Wales. Neste *dataset*, constam as informações relevantes para o estudo, para o treinamento de um modelo de *Deep Learning* utilizando uma rede neural convolucional a ponto de torná-lo capaz de classificar corretamente quando uma comunicação é um ataque ou não.

Visando a capacidade de identificação de ataques em tempo real, o modelo deve

ser aplicável a um sistema embarcado, para que a inferência possa ocorrer no mesmo dispositivo em que os dados sobre a rede são captados. Esse critério leva em consideração minimizar a latência no envio de dados para uma outra unidade computacional mais robusta. Portanto, esta obra visa o desenvolvimento de um algoritmo de rede neural de classificação apto a identificar ataques em uma rede sem fio e ainda capaz de ser adaptado para TinyML, tornando-o aplicável para ambientes com recursos computacionais limitados.

## 3.2 Definição do *dataset*

Diante do acervo na literatura de trabalhos acerca da temática de cibersegurança, dentre os materiais utilizados destaca-se o *dataset* desenvolvido por Nour Moustafa [16] pela *University of New South Wales*, na Austrália, UNSW-NB15 que foi desenvolvido com o objetivo de criar um conjunto de dados útil para o estudo da segurança de redes, mas que se diferenciasse dos *datasets* já existentes por trazer em sua composição atributos presentes nas comunicações atuais.

O UNSW-NB15 está disponível no site da universidade[1] e é composto por quatro arquivos em formato CSV que são quatro partes de uma única base de dados. Além desses arquivos foi utilizado outro CSV, também disponibilizado no site, no qual estão todas as *features* do *dataset*, bem como as descrições de cada uma.

A base de dados abrange a identificação de nove ataques familiares, a saber:

1. *Fuzzers*;
2. *Analysis*;
3. *Backdoors*;
4. DoS;
5. *Exploits*;
6. Genérico;
7. *Reconnaissance*;

[1]Este *dataset* está disponível em `https://research.unsw.edu.au/projects/unsw-nb15-dataset`

8. *Shellcode*;

9. *Worms*

No total são 49 atributos, considerando o rótulo de classificação, e 2.540.044 registros com o rótulo variando entre `attack` ou `normal`. Para acessar o arquivo foi utilizando a linguagem de programação ***Python*** e a biblioteca **Pandas**. Todo o conjunto de dados foi unido em um único *DataFrame* para que houvesse a divisão entre a variável alvo (variável a qual o modelo deve aprender a classificar), denominada $y$, dos demais atributos, conjunto $X$.

## 3.3 Análise de dados

As *features* do *dataset* contém valores numéricos e nominais, sendo 39 do primeiro caso e 9 do segundo (não incluindo a variável alvo que é do tipo nominal). A primeira etapa da análise foi buscar limpar o conjunto de dados, removendo atributos, lidando com elementos ausentes e do tipo *Not a Number* (NaN), observando a distribuição dos dados para tomadas de decisão posteriores.

### 3.3.1 Atributos categóricos

Dentro do *dataset*, alguns atributos categóricos se referem a endereços, portas e tempo de conexão, os quais possuem uma grande diversidade dentre os valores que o atributo pode assumir, não podendo delimitá-los dentro de um intervalo ou de algumas categorias. Diante disso, a etapa de rotulação de variável categórica era inviável dado o número altíssimo de categorias que deveriam ser codificadas para valores numéricos. Portanto, esses atributos foram removidos dentre os dados.

Além deles, a categoria `attack_cat`, que traz consigo a informação de qual o tipo de ataque identificado, foi igualmente removida, tendo em vista a enorme quantidade de valores ausentes nessa categoria, remover as linhas que possuíssem esse caso não seria uma opção pelo fato de que reduziria o *dataset* em 99,126%, restando 22.215 registros para o treinamento.

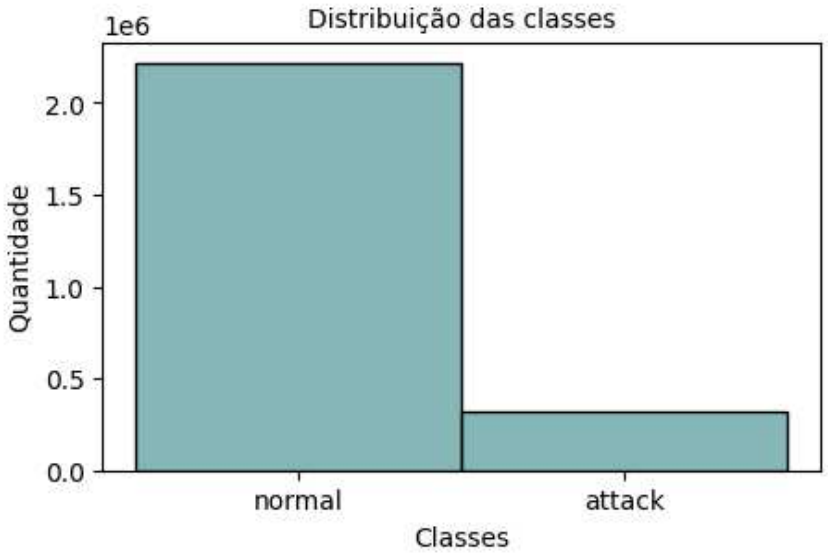
### 3.3.2 Elementos ausentes e NaNs

Tendo removido sete atributos categóricos, o *dataset* continua apresentando ausências e valores do tipo NaN em sua composição. Para contornar isso foi utilizada a técnica de imputação de valores através do método `SimpleImputer` da biblioteca `sklearn` utilizando o argumento `strategy= most_frequent`.

### 3.3.3 Distribuição dos dados

O *dataset* possui um número de registros rotulados como `normal` consideravelmente maior que o de registros rotulados como `attack`. Esse fato é apresentado na figura a seguir. São 2.218.760 registros do tipo `normal` e 321.283 registros de `attack`, o que corresponde a uma divisão de, aproximadamente, 12,65% dos dados sendo de ataques.

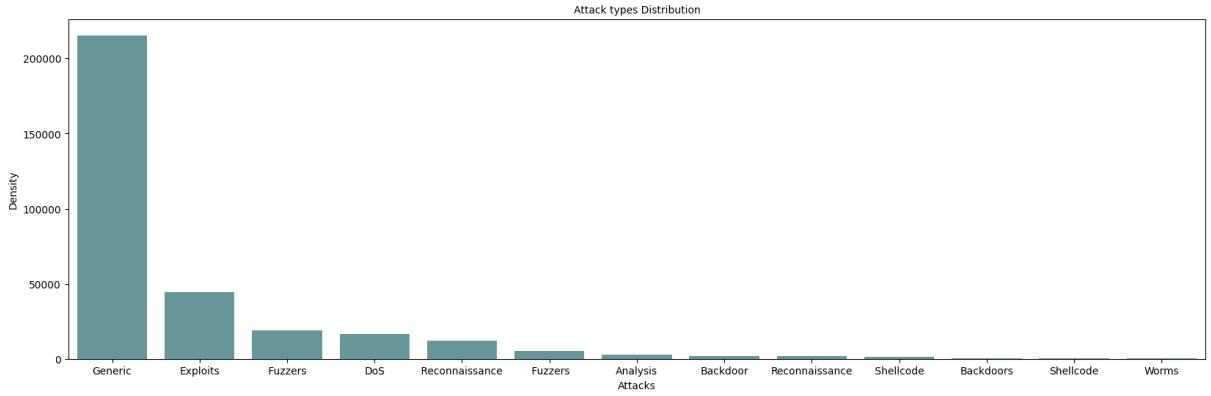
Figura 3.1: Distribuição das classes no *dataset*




Fonte: Elaborada pelo autor

Dentre os ataques, os seus tipos são distribuídos em 13 classes, contudo com a maior concentração deles no tipo `Generic`. O histograma abaixo apresenta a quantidade de cada tipo de ataque presente no dataset.

Figura 3.2: Distribuição dos tipos de ataques




Fonte: Elaborada pelo autor

## 3.4 Pré-processamento

### 3.4.1 LabelEncoder

Modelos de aprendizagem de máquina são baseados em computação, portanto, processam dígitos e não possuem a capacidade de entender uma palavra em sua forma original. Para lidar com esse fator, técnicas de codificação de categorias são utilizadas. Dentre as opções disponíveis, o `LabelEncoder` foi escolhido por codificar variáveis categóricas que não possuem hierarquia entre elas. Todas as variáveis nominais que ainda haviam no *dataset* foram codificadas por esse codificador pertencente à biblioteca `sklearn`.

### 3.4.2 Divisão entre dados de treino e dados de teste

Tendo mantido os atributos considerados pertinentes e convertido todo o *dataset* para valores numéricos, o conjunto total foi separado em três subconjuntos: treino com 70% do *dataset*, teste com 15% e validação com 15%. Foi verificado o balanceamento das classes da variável alvo em cada subconjunto e o resultado foi:

Tabela 3.1: Distribuição das classes `normal` e `attack` nos conjuntos de dados

| | `normal` | `attack` | razão |
|---|---|---|---|
| **treino** | 1.553.480 | 224.550 | 14,45% |
| **teste** | 332.626 | 48.380 | 14,54% |
| **validação** | 332.654 | 48.353 | 14,54% |

Os dados de treino foram utilizados para todos os modelos escolhidos, os dados de teste foram utilizados para verificação do quão efetivo cada modelo estava após o treinamento e os dados de validação foram reservados para a verificação da performance do modelo após convertido em TinyML.

## 3.5 Definição de modelos

Foram definidos três modelos a serem treinados: *Random Forest*, *Decision Forest*, Rede Neural (`Sequential` da plataforma TensorFlow). Todos os modelos foram treinados pela plataforma do Google Colaboratory Pro, usando de um ambiente de execução com 334,56GB de RAM e 225,33GB de Disco, além do uso de TPUs[2] para acelerar o processo.

### 3.5.1 *Random Forest*

O modelo `RandomForestClassifier` foi treinado usando as seguintes definições:

Tabela 3.2: Definição dos parâmetros do modelo `RandomForestClassifier`

| | |
|---|---|
| `n_estimators` | 100 |
| `random_state` | 1 |

O tempo para treinamento desse modelo foi de 4 minutos.

### 3.5.2 Tensorflow *Decision Forest*

Considerando a posterior adaptação do modelo para TinyML, foi utilizado um modelo que possui o princípio de funcionamento equivalente ao `RandomForestClassifier`, porém implementado em TensorFlow, pois nessa plataforma existem facilidades para a conversão de modelos pela função `TFLiteConverter`, que adapta o modelo treinado para `TFLite`[3].

Utilizando o *framework* Keras, o modelo `keras.RandomForestModel` foi treinado utilizando a configurações padrão dele, sem nenhum parâmetro sendo explicitamente definido. O tempo para o treinamento desse modelo foi de 59 segundos.

[2]**TPU**: unidade de processamento de tensor, do inglês *Tensor Processing Unit*

[3]O **TFLite** é um conjunto de ferramentas para aprendizado de máquina que possibilita a execução de modelos em dispositivos IoT

### 3.5.3 Rede Neural Convolucional

Para o treinamento de um modelo dessa categoria, uma adaptação do formato dos dados foi feita, convertendo todos os tipos das variáveis para o tipo `np.float32`. Com isso, todos os dados foram transformados em tensores, pois a rede neural desenvolvida utiliza esse tipo de formato para seus dados de entrada.

Os hiperparâmetros definidos para o modelo `Sequential` foram:

Tabela 3.3: Hiperparâmetros da rede neural

| Hiperparâmetro | Valor |
|---|---|
| `Dense` | 128 |
| `Dropout` | 0,4 |
| `Dense` | 10 |
| `learning_rate` | 0,01 e 0,001 |
| `Epoch` | 10 |

Como funções de ativação foram utilizadas a ReLU e a Softmax nas camadas `Dense` de 128 conexões e 10 conexões, respectivamente. Para cada taxa de aprendizado (*learning_rate*) foram executadas dez épocas para treinamento da rede neural. Como otimizador, foi utilizado o Gradiente Descendente. O modelo, considerando o treinamento para as duas `learning_rates` levou 28 minutos para concluir o treinamento e mais 13 minutos para treinar usando o otimizador gradiente descendente.

# Capítulo 4

# Resultados Obtidos

Neste capítulo são apresentados os resultados, a partir da observação de métricas, de todos os modelos definidos.

## 4.1 Avaliação dos modelos

A avaliação dos modelos se baseia na utilização das métricas de avaliação: Acurácia, Precisão, *Recall*, *F1-Score*, *Mean Absolute Error*. Todos os modelos foram treinados no mesmo ambiente sob as mesmas disposições de recursos de TPU, RAM e CPU. Vale ressaltar que, para os modelos baseados em TensorFlow, o formato de arquivo utilizado para o treinamento não foi o mesmo que o RandomForest utilizou. Os vetores de dados necessitaram ser convertidos para um formato compatível com o tipo de dados esperados na definição dos modelos. Todavia, a dimensão dos vetores foi preservada. Os resultados são apresentados na tabela 4.1.

O modelo que mais se destacou nos resultados foi o RandomForest, que inclusive chegou a 100% nos primeiros testes em um cenário que todas as linhas que possuíam valores NaN foram removidas. Contudo, mesmo depois da etapa de análise de dados, fazendo um tratamento mais adequado (imputando valores ausentes e convertendo variáveis categóricas) os resultados em cada métrica permaneceram altos, os mais próximos ao 100%. Esse fato foi alvo de questionamento, contudo, em fóruns de desenvolvedores foram vistos relatos de que valores assim são comuns na etapa de treinamento dessa classe de modelo.

Além dos resultados em cada métrica, outro fator que chamou atenção foi o tempo de treinamento. Estes dados foram agrupados e apresentados na tabela 4.2:

Tabela 4.1: Resultados da avaliação de cada modelo

| | Acurácia | Precisão | *Recall* | *F1-Score* | MAE |
|---|---|---|---|---|---|
| **Random Forest** | 99,45% | 98,04% | 97,65% | 97,84% | 0,0055 |
| **Tensorflow Decision Forest** | 87,30% | 76,22% | 87,30% | 81,38% | 0,1270 |
| **Rede Neural Convolucional** | 87,30% | 76,22% | 87,30% | 81,38% | 0,1269 |

Tabela 4.2: Tempo de treinamento de cada modelo

| Modelo | Tempo de treinamento (segundos) |
|---|---|
| **Random Forest** | 240 |
| **Tensorflow Decision Forest** | 59 |
| **Rede Neural Convolucional** | 2460 |

Tanto o modelo de `RandomForest` como o de `DecisionForest` são construídos sobre o mesmo princípio de funcionamento: são um conjunto de árvores de decisão, onde cada árvore é uma estrutura hierárquica de nós e folhas. Cada nó interno representa um teste em um atributo, e as folhas representam as classes ou valores previstos. Contudo, o *framework* em que cada um dos modelos foi desenvolvido é diferente. O primeiro foi implementado pela biblioteca `sklearn`, módulo de *machine learning* com métodos otimizados, enquanto o segundo foi desenvolvido no TensorFlow, que pode se utilizar de GPUs, quando disponíveis, para aceleração dos cálculos no processo.

Diante desses fatores, a diferença de tempo de treinamento entre eles se torna plausível. Entretanto, o mesmo raciocínio não se aplica ao terceiro modelo pelo fato de que ele não segue o mesmo funcionamento, antes, é uma estrutura complexa formada por camadas de perceptrons. A complexidade e o número de épocas utilizadas para o treinamento da Rede Neural, são refletidos no maior tempo demandado para treiná-lo.

## 4.2 Conversão para TFLite

A partir do que é verificado na avaliação dos modelos, o modelo RandomForestClassifier se apresenta como a melhor opção para o problema, devido aos seus bons resultados

e um tempo de treinamento ainda justo. Todavia, a conversão de um modelo da biblioteca `sklearn` para `TFLite` não foi possível durante a execução desse trabalho por ainda não existir um conversor eficiente para tal tarefa. Sendo assim, apesar de ter sido o modelo de melhor performance, não atende às necessidades prioritárias de poder ser compatível com um ambiente de TinyML.

Por conseguinte, o modelo do próprio Tensorflow foi buscado como uma alternativa de executar um modelo cuja performance seja tão boa quanto o do `sklearn`, pois seu princípio de funcionamento é semelhante, e por ser nativo da plataforma possibilitasse sua conversão para `TFLite`. Entretanto, o Tensorflow ainda não possui métodos que implementam a conversão de um modelo de *Decision Forest* aplicável para TinyML.

Sendo assim, o modelo de rede neural foi buscado tendo em mente o critério de conversão em um modelo `TFLite` que, nesse caso, é atendido pela biblioteca do Tensorflow por meio do método `TFLiteConverter`. Sendo assim, o modelo de rede neural foi escolhido e convertido como modelo `TFLite`.

## 4.3 Validação em contexto de TinyML

Com o modelo definido, ele foi carregado em um ambiente de máquina virtual Linux, na distribuição Ubuntu, no qual foi desenvolvido um algoritmo que carrega modelos do tipo `TFLite`, lê os arquivos de validação separados anteriormente, e executa as predições. Os resultados obtidos pelo modelo de TinyML foram:

Tabela 4.3: Resultados da avaliação do modelo `TFLite`

| | Acurácia | Precisão | *Recall* | *F1-Score* | MAE |
|---|---|---|---|---|---|
| **Rede Neural TFLite** | 87,31% | 76,23% | 87,31% | 81,39% | 0,1269 |

# Capítulo 5

# Considerações finais

Diante dos resultados obtidos nesse trabalho, foi verificado que a conversão do modelo desenvolvido se manteve igualmente eficiente mesmo quando restrito a um ambiente de TinyML utilizando o `TFLite`. Esse fator exemplifica a capacidade dos algoritmos atuais de aprendizagem de máquina de conseguir reproduzir, mesmo em meio a restrições, sem muitas perdas de qualidade modelos robustos. Ademais, a utilização das métricas foi essencial para uma análise quantitativa das avaliações, servindo como base sólida para comparações e para análise de modelos.

O modelo proposto atingiu uma taxa de acerto (considerando a acurácia) satisfatória de 87,30% e mantendo-a ao converter o modelo ao `TFLite`. Essa métrica poderia ser ainda melhorada através de outras abordagens e metodologias como a normalização dos dados, otimização de hiperparâmetros usando, por exemplo, o *GridSearch*, entre ainda outros fatores. Esse valor atingido se mostra aceitável dado as restrições de tempo durante o desenvolvimento desse trabalho e desafios imprevistos durante o percurso.

Conclui-se, portanto, que a utilização de um modelo de CNN é capaz de ser aplicado em um contexto diferente do qual é comumente empregado (reconhecimento de imagens) e se mostra eficiente para classificar ataques em redes cibernéticas.

## 5.1 Trabalhos futuros

Inicialmente, foi planejado a importação do modelo de TinyML para um sistema cujas restrições computacionais são reais. Foi definido o sistema embarcado Raspberry Pi Model 3B+, para que o modelo convertido fosse executado nele e realizasse as inferências

a partir do conjunto de dados de validação. Contudo, devido a impedimentos na gestão de bibliotecas externas, o uso do `tflite-runtime` custou mais tempo do que o esperado, além de que a versão de alguns módulos era incompatível com a versão na qual o tflite-runtime foi desenvolvido.

Por conta disso, a alternativa para ainda serem obtidos dados através do tflite-runtime e a obtenção de resultados por um modelo de TinyML, foi utilizar uma máquina virtual Linux. Todavia, ainda está no horizonte de trabalhos a serem feitos a real adaptação do modelo em um sistema embarcado, o que traria mais praticidade pelo fato do embarcado poder ser inserido em diferentes redes.

Algumas outras atividades, originalmente planejadas, não puderam ser desenvolvidas devido a fatores que as impediram. Todavia, também são próximos passos a serem tomados para o aperfeiçoamento desse trabalho:

- Normalização dos dados;
- Otimização de hiperparâmetros;
- Monitoramento de uma rede local para leitura de dados de rede reais;
- Adaptação para que as classificações sejam realizadas em tempo real no passo em que o monitoramento é feito.

# Referências Bibliográficas

[1] KUROSE, J.; ROSS, K. **Computer Networking: A Top-Down Approach**. 6. ed. Boston: Pearson, 2016.

[2] GeeksforGeeks. **User Datagram Protocol (UDP)**. Disponível em: `https://www.geeksforgeeks.org/user-datagram-protocol-udp/`. Acesso em: 07 de outubro de 2024.

[3] STALLINGS, William. **Data and computer communications**. 8. ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2006. ISBN 0132433109.

[4] REDDI, V. J. **Machine Learning Systems: Principles and Practices of Engineering Artificially Intelligent Systems**. Harvard University, 2024. Disponível em: `https://mlsysbook.ai/`. Acesso em 14 de outubro de 2024.

[5] RUSSELL, Stuart; NORVIG, Peter. **Artificial Intelligence: A Modern Approach**. 4. ed. Harlow, Essex: Pearson, 2020.

[6] CLOUDFLARE. **Internet Control Message Protocol (ICMP)**. Disponível em: `https://www.cloudflare.com/learning/ddos/glossary/internet-control-message-protocol-icmp/`. Acesso em: 07 out. 2024.

[7] FULIN, Xiang. **ICMP (Internet Control Message Protocol)**. Disponível em: `https://info.support.huawei.com/info-finder/encyclopedia/en/ICMP.html`. Acesso em: 07 out. 2024.

[8] IBM. Disponível em: `https://www.ibm.com/br-pt/topics/malware`. Acesso em: 09 de outubro de 2024.

[9] Moore, D., Voelker, G., Savage, S. Inferring Internet Denial of Service Activity. In: Proc. 2001 USENIX Security Symposium. Washington, DC, Aug. 2001.

[10] IBM. **What is network monitoring?**. Disponível em: https://www.ibm.com/topics/network-monitoring. Acesso em: 09 de outubro de 2024.

[11] COSTA, Marcelo Jorge de Carvalho; CAMARGO, Celso Pereira de Souza. **A importância do aprendizado ativo na formação médica: uma revisão integrativa**. Revista Brasileira de Educação Médica, São Paulo, v. 44, n. 2, p. e067, 2020. Disponível em: https://www.scielo.br/j/rbem/a/CmX7jQCmqp7nsns3QnfWqJS/?format=pdf&lang=pt. Acesso em: 15 out. 2024.

[12] SARKER, Iqbal H.; FURHAD, Md Hasan; NOWROZY, Raiyan. **AI-driven cybersecurity: an overview, security intelligence modeling and research directions**. SN Computer Science, [S.l.], v. 2, n. 3, p. 173, 2021. Disponível em: https://doi.org/10.1007/s42979-021-00564-y. Acesso em: 15 out. 2024.

[13] HERATH, Sandaruwan. **Theoretical basis of ML model evaluation metrics: summary**. Medium, mar. 2024. Disponível em: https://medium.com/image-processing-with-python/theoretical-basis-of-ml-model-evaluation-metrics-summary-3cae19129679. Acesso em: 16 out. 2024.

[14] AHMED, M Waqar. **Understanding Mean Absolute Error (MAE) in Regression: A Practical Guide**. Medium, aug. 2023. Disponível em: https://medium.com/@m.waqar.ahmed/understanding-mean-absolute-error-mae-in-regression-a-practical-guide-26e80ebb
Acesso em: 16 out. 2024.

[15] ALTUNAY, H. C.; ALBAYRAK, Z. A hybrid CNN-LSTM intrusion detection system for industrial IoT networks. *Engineering Science and Technology, an International Journal*, v. 38, p. 101322, 2023. DOI: 10.1016/j.jestch.2022.101322.

[16] MOUSTAFA, Nour; SLAY, Jill. **UNSW-NB15: A comprehensive data set for network intrusion detection systems**. Military Communications and Information Systems Conference (MilCIS), Canberra, 2015. p. 1-6. Disponível em: https://doi.org/10.1109/MilCIS.2015.7348942. Acesso em: 16 out. 2024.

[17] FORBES. **Internet Usage Statistics In 2024**. Forbes, 2024. Disponível em:

https://www.forbes.com/home-improvement/internet/internet-statistics/. Acesso em: 23 out. 2024.

[18] PAYÃO, Felipe. **Banco Inter é extorquido e dados de clientes são expostos; invasão é negada**. TecMundo, 04 maio 2018. Disponível em: https://www.tecmundo.com.br/seguranca/129811-exclusivo-vazam-dados-400-mil-clientes-banco-inter.htm. Acesso em: 23 out. 2024.