UNIVERSIDADE ESTADUAL PAULISTA

"JÚLIO DE MESQUITA FILHO"

Câmpus Bauru

CARLOS DE JESUS REIS

FEDERATED TRANSFER LEARNING PARA DETECÇÃO DE INTRUSÃO EM REDES DE COMPUTADORES

BAURU

Março/2025

CARLOS DE JESUS REIS

FEDERATED TRANSFER LEARNING PARA DETECÇÃO DE INTRUSÃO EM REDES DE COMPUTADORES

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Instituto de Biociências, Letras e Ciências Exatas, Campus Bauru.
Orientador: Prof. Dr. Kelton Costa

BAURU
Março/2025

R375f

Reis, Carlos de Jesus
Federated Transfer Learning para Detecção de Intrusão em Redes de Computadores. / Carlos de Jesus Reis. -- Bauru, 2025
103 p.

Dissertação (Mestrado) - Universidade Estadual Paulista (UNESP), Faculdade de Ciências, Bauru
Orientadora: Kelton Augusto Pontara Costa

1. Federated Learning. 2. Transfer Learning. 3. Federated Transfer Learning. 4. Intrusion Detection System. 5. Machine learning. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

ATA DA DEFESA PÚBLICA DA DISSERTAÇÃO DE MESTRADO DE CARLOS DE JESUS REIS, DISCENTE DO PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO, DA FACULDADE DE CIÊNCIAS - CÂMPUS DE BAURU.

Aos 07 dias do mês de março do ano de 2025, às 14h, por meio de Videoconferência, realizou-se a defesa de DISSERTAÇÃO DE MESTRADO de CARLOS DE JESUS REIS, intitulada **FEDERATED TRANSFER LEARNING PARA DETECÇÃO DE INTRUSÃO EM REDES DE COMPUTADORES**. A Comissão Examinadora foi constituída pelos seguintes membros: Prof. Dr. KELTON AUGUSTO PONTARA DA COSTA (Orientador(a) - Participação Virtual) do(a) FC / UNESPBauru SP, Prof. Dr. LEANDRO APARECIDO PASSOS JUNIOR (Participação Virtual) do(a) Pós-doutorando do Depto. de Computação / FC/Bauru - Unesp, Prof. Dr. ALEX MARINO GONÇALVES DE ALMEIDA (Participação Virtual) do(a) Computação / Faculdade de Tecnologia de Ourinhos. Após a exposição pelo mestrando e arguição pelos membros da Comissão Examinadora que participaram do ato, de forma presencial e/ou virtual, o discente recebeu o conceito final APROVADO. Nada mais havendo, foi lavrada a presente ata, que após lida e aprovada, foi assinada pelo(a) Presidente(a) da Comissão Examinadora.

Prof. Dr. KELTON AUGUSTO PONTARA DA COSTA

Carlos de Jesus Reis

Federated Transfer Learning para Detecção de Intrusão em Redes de Computadores

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Instituto de Biociências, Letras e Ciências Exatas, Campus Bauru.

Banca Examinadora

Prof. Dr. Kelton Costa
Orientador

Prof. Dr. Leandro Aparecido Passos Junior
UNESP - Câmpus de Bauru

Prof. Dr. Alex Marino Gonçalves de Almeida
Faculdade de Tecnologia de Ourinhos

São José do Rio Preto, 07 de março de 2025.

Agradecimentos

A jornada de construção deste trabalho foi repleta de desafios, aprendizados e superações, e nada disso teria sido possível sem o apoio, primeiramente, de Deus — que me faz ser resiliente e persistente — e também de pessoas fundamentais na minha vida.

Agradeço, com profunda gratidão, aos meus pais José Macedônio e Alcidia, com sua simplicidade e sabedoria, conseguiram me ensinar o valor do esforço, da honestidade e da dedicação. Foram e continuam sendo meu alicerce, minha base e minha inspiração diária.

À minha irmã Lia, pelo carinho, incentivo e apoio constante em todos os momentos. Sua presença me trouxe leveza e força nos períodos mais intensos desta caminhada.

À minha esposa Renata e aos meus filhos João Pedro e Maria Eduarda, pela paciência, compreensão e amor incondicional durante todas as etapas deste processo. Vocês são minha motivação maior e minha razão de perseverar.

Ao meu orientador, professor Kelton, pela orientação precisa, pela disponibilidade e pelas contribuições valiosas que foram determinantes para o desenvolvimento desta dissertação. Sua experiência e dedicação foram fontes essenciais de aprendizado e ao meu cordenador Thiago com sua objetividade e conhecimento.

Agradeço também aos amigos e colaboradores Carlos Tojeiro e Anselmo Araújo, pelo companheirismo, pelas discussões construtivas e por sempre estarem dispostos a contribuir de forma generosa com ideias, revisões e sugestões ao longo do projeto.

A todos que, direta ou indiretamente, contribuíram para a realização deste trabalho, deixo aqui meu mais sincero e profundo agradecimento.

Agradeço a Deus por ter obtido mais um conquista importante. A caminhada é difícil, mas com fé alcançamos tudo aquilo que Ele nos permite.

Resumo

Com a crescente inserção da tecnologia da informação em diversos processos do mundo atual, observa-se um aumento significativo nos ciberataques a infraestruturas de redes e dispositivos. A propagação de *softwares* maliciosos, capazes de comprometer a integridade de diferentes sistemas, tem impulsionado empresas a investirem continuamente em segurança cibernética. Nesse contexto, as técnicas de *Machine Learning* (ML) vêm sendo cada vez mais utilizadas na cibersegurança, permitindo a criação de Sistemas de Detecção de Intrusão (IDS). No entanto, a maioria dos estudos e implementações existentes adotam uma abordagem centralizada, em que as atividades de treinamento e teste ocorrem exclusivamente em um servidor central, limitando a escalabilidade e aumentando os riscos associados ao compartilhamento de dados sensíveis. Diante desse cenário, este trabalho propõe a aplicação de *Federated Learning* (FL) aliado ao *Transfer Learning* (TL), resultando em uma arquitetura de *Federated Transfer Learning* (FTL). O FL permite a execução descentralizada e colaborativa do treinamento e teste dos modelos, garantindo a privacidade dos dados ao evitar sua transmissão para um servidor central. Além disso, essa abordagem mitiga os desafios relacionados à transferência de grandes volumes de dados. O TL, por sua vez, possibilita reaproveitar o conhecimento previamente adquirido, otimizando o desempenho dos modelos em diferentes cenários. A orquestração do FL foi realizada por meio do *Framework Flower*, amplamente utilizado no meio acadêmico, enquanto a gestão da TL foi conduzida por uma *pipeline* de *DevOps*, alternativa inovadora para automatizar e garantir a segurança do processo de transferência de modelos pré-treinados. Para validar a proposta, foram utilizados os conjuntos de dados BOT-IoT e TON_IoT, que contêm amostras rotuladas de tráfego de rede. O estudo utilizou uma *Linear Neural Networks* (LNN) como modelo compartilhado e o algoritmo de agregação *FedAvg* para o treinamento descentralizado. Os experimentos demonstraram resultados excepcionais em todas as comparações realizadas, evidenciando a eficácia da abordagem híbrida entre Aprendizado Federado e Aprendizado por Transferência na detecção e mitigação de intrusões. A pesquisa contribui significativamente para o avanço das soluções de cibersegurança baseadas em Inteligência Artificial, incentivando novas investigações sobre técnicas de agregação e aprimoramento da arquitetura FTL para ambientes IoT e *Edge Computing*.

Palavras-chave: Federated Learning, Transfer Learning, Federated Transfer Learning, Intrusion Detection System, Network Intrusion Detection System, Machine learning

Abstract

With the growing integration of information technology in various processes of today's world, there is a significant increase in cyberattacks on network infrastructures and devices. The spread of malicious software, capable of compromising the integrity of different systems, has driven companies to continuously invest in cybersecurity. In this context, Machine Learning (ML) techniques are increasingly being used in cybersecurity, enabling the creation of Intrusion Detection Systems (IDS). However, most existing studies and implementations adopt a centralized approach, where training and testing activities occur exclusively on a central server, limiting scalability and increasing risks associated with sharing sensitive data. Against this backdrop, this work proposes the application of Federated Learning (FL) combined with Transfer Learning (TL), resulting in a Federated Transfer Learning (FTL) architecture. FL allows decentralized and collaborative training and testing of models, ensuring data privacy by preventing its transmission to a central server. Furthermore, this approach mitigates challenges related to transferring large volumes of data. TL, in turn, enables the reuse of previously acquired knowledge, optimizing model performance in different scenarios. The orchestration of FL was carried out through the Flower framework, widely used in academia, while the management of TL was conducted by a DevOps pipeline, an innovative alternative to automate and ensure the security of the pre-trained model transfer process. To validate the proposal, the BOT-IoT and TON_IoT datasets, which contain labeled samples of network traffic, were used. The study utilized a linear artificial neural network (LNN) as the shared model and the FedAvg aggregation algorithm for decentralized training. The experiments demonstrated exceptional results across all comparisons made, highlighting the effectiveness of the hybrid approach between Federated Learning and Transfer Learning in intrusion detection and mitigation. The research significantly contributes to the advancement of AI-based cybersecurity solutions, encouraging further investigations into aggregation techniques and improvements to the FTL architecture for IoT and Edge Computing environments.

Palavras-chave: Federated Learning, Transfer Learning, Federated Transfer Learning, Intrusion Detection System, Network Intrusion Detection System, Machine learning

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ACC - *Accuracy (Acurácia)*

AUC - *Area Under the Curve*

CNN - *Convolutional neural network*

CNN-LSTM - *Long Short-Term Memory architecture fitted with Convolutional Neural Network in its hidden layers*

DANN - *Direct Artificial Neural Networks*

DIoT - *Sistema de autoaprendizagem para detectar dispositivos comprometidos em redes*

DL - *Deep Learning*

DNN - *Deep Neural Network*

DR - *Taxa de detecção*

DT - *Decision Tree*

EoT - *Edge of Things*

FDL - *Federated Deep Learning*

FDNN - *Federated Deep Neural Network*

FedAGRU - *Unidade Recorrente Gated Attention Gated*

FedAvg - *Federated averaging*

FL - *Federated Learning*

FPR - *Taxa de falsa predição*

FTL - *Federated Transfer Learning* item[GRU]- *Unidade Recorrente Controlada por Porta*

IA - *Inteligência Artificial*

ICV - *Veículos Conectados Inteligentes*

IDS - *Intrusion Detection System*

IIoT - *Industrial Internet of Things*

IoT - *Internet of things*

HDIS - *Host Intrusion Detection System*

LNN - *Linear Neural Networks*

LR - *Logistic Regression*

MEC - *Mobile Edge Computing*

ML - *Machine Learning*

MLP - *Multi Layer Perception*

NB - *Naive Bayes*

NBTree - *Naive Bayesian Tree*

NIDS - *Network Intrusion Detection System*

NILM - *Non-intrusive Load Monitoring*

RF - *Random Forest*

RNN - *Recurrent Neural Network*

RT - *Random Tree*

STIN - *Redes Integradas Satélite-Terrestre*

SVM - *Support Vector Machine*

TL - *Transfer Learning*

UDP - *User Datagram Protocol*

WENs - *Wireless Edge Networks*

KVM - *Máquina Virtual baseada em Kernel*

Sumário

1 Introdução

As redes de computadores, os sistemas distribuídos e a Internet são componentes fundamentais para o desenvolvimento da sociedade. O avanço constante das tecnologias digitais e a crescente interconectividade de dispositivos nos diversos setores abastecidos pela tecnologia têm trazido significativas contribuições para a economia, impactando diretamente o trabalho e o estilo de vida das pessoas (GU; LU, 2021).

Além de representarem a base da sociedade moderna, os recursos computacionais, redes, sistemas e a própria Internet armazenam grandes volumes de dados, incluindo informações pessoais e sensíveis, como históricos de saúde e transações financeiras. Essas informações tornaram-se acessíveis por meio das infraestruturas tecnológicas corporativas. O mercado varejista, por sua vez, identificou uma grande oportunidade para expandir seus negócios por meio da interação on-line com os clientes, o que impulsionou significativamente o crescimento das plataformas de e-commerce.

Contudo, essa crescente digitalização também trouxe riscos. Países, especialmente as nações desenvolvidas, tornaram-se alvos frequentes de ataques promovidos por crackers (STERLE; BHUNIA, 2021), uma vez que informações estratégicas desses países possuem relevância no contexto geopolítico e na competitividade econômica.

Com o aumento da integração e dependência desses recursos tecnológicos, a segurança da informação emergiu como uma preocupação central em todos os setores. Isso se deve ao crescimento exponencial no número de ataques cibernéticos. Nesse cenário, os *Intrusion detection System* (IDS) desempenham um papel crucial na proteção de redes e sistemas contra atividades maliciosas, sendo responsáveis por preservar a integridade e a confiabilidade das informações (OSKEN et al., 2019).

Destacam-se, ainda, dois pontos importantes evidenciados no Relatório de Custo da Violação de Dados de 2022, da IBM Security. O primeiro refere-se ao aumento do custo médio total de uma violação de dados, que passou de US$ 4,24 milhões para US$ 4,35 milhões. O segundo ponto mostra que empresas que adotaram soluções baseadas em inteligência artificial (IA) e automação para segurança apresentaram uma redução significativa desses custos, chegando a uma economia de até US$ 3,05 milhões quando comparadas a empresas que não implementaram tais recursos (IBM, 2022).

Com base no levantamento realizado entre janeiro de 2020 e janeiro de 2025, utilizando dados disponibilizados pelo Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil (CERT.br) (ESTUDOS, 2025), observa-se um panorama relevante acerca dos principais tipos de ataques cibernéticos, que possuem relação direta com a presente pesquisa. Dentre os ataques mais recorrentes, destacam-se: Negação de Serviço (DoS), Fraude, Invasão,

Varredura (*Scan*) e ataques à Web.

No período analisado, o número total de incidentes manteve-se dentro de um intervalo de 457 mil a 630 mil ocorrências, evidenciando a persistência das ameaças cibernéticas no cenário nacional. Em especial, chama atenção o crescimento expressivo dos ataques classificados como *Scan*, que buscam mapear redes e identificar vulnerabilidades exploráveis em sistemas e dispositivos conectados. No ano de 2024, tais ataques corresponderam a aproximadamente 80% do total registrado, configurando-se como a principal ameaça enfrentada.

Nos últimos anos, observou-se também o avanço do uso de técnicas como aprendizado de máquina, aprendizado profundo e inteligência artificial aplicadas aos IDS. Diversas abordagens vêm sendo incorporadas aos sistemas com o intuito de melhorar os indicadores de desempenho e eficácia na detecção de ameaças.

Nesse contexto, os IDS assumem um papel essencial para a segurança da informação, exigindo a aplicação de um conjunto de técnicas e conhecimentos especializados para sua efetividade. Para fundamentar essa abordagem, realizou-se uma revisão teórica detalhada, incluindo trabalhos correlatos e referências bibliográficas, proporcionando uma base sólida para a compreensão do tema. Essa fundamentação possibilita uma análise aprofundada das principais abordagens, algoritmos e técnicas relevantes, oferecendo insights valiosos. Com base nesse estudo, tornou-se possível avaliar o potencial de aplicação dessas abordagens no desenvolvimento de IDS mais eficazes, seguros e adaptáveis a ambientes distribuídos e heterogêneos.

O objetivo geral desta dissertação consistiu em desenvolver e apresentar uma arquitetura inovadora para detecção de intrusão, utilizando a combinação de duas técnicas de *Machine Learning* (ML) que têm se mostrado bastante promissoras: o *Federated Learning* (FL) e o *Transfer Learning* (TL). Essa integração resultou no uso do *Federated Transfer Learning* (FTL), proposta com o intuito de aumentar a eficiência e escalabilidade dos IDS, considerando a crescente complexidade e volume dos ataques cibernéticos, sem comprometer a privacidade dos dados dos usuários.

A utilização do FL permite o treinamento colaborativo dos modelos de aprendizado sem a necessidade de centralizar os dados, assegurando assim a proteção das informações sensíveis. Simultaneamente, a incorporação do TL possibilita o reaproveitamento do conhecimento adquirido previamente em cenários similares, otimizando o desempenho dos modelos em novos contextos e reduzindo o tempo e os custos de treinamento.

Os Objetivos Específicos que se apresentam, são:

A - Apresentação do Tema e Justificativa da Pesquisa:

- Introduzir o tema da pesquisa, destacando sua relevância e justificando sua escolha como proposta de dissertação;

- Explicar a importância do estudo no contexto atual da segurança cibernética e sua contribuição acadêmica e prática.

B - Realizar uma Revisão Sistemática da Literatura:

- Demostrar o estado-da-arte através da apresentação de uma Revisão Sistemática da Literatura, descrita na subseção 2.1;
- Coletar e analisar artigos relevantes sobre detecção de intrusão, FL e TL nos últimos cinco anos;
- Identificar melhores práticas, conjuntos de dados e arquiteturas utilizadas em estudos semelhantes.

C - Selecionar recursos e conjuntos de dados:

- Escolher conjuntos de dados representativos para detecção de intrusão, como TON_IoT e BOT-IoT;

D - Desenvolver o modelo e a arquitetura

- Definir a arquitetura do modelo, utilizando uma *Linear Neural Networks* (LNN) como modelo global, que possa ser utilizado de forma centralizada e descentralizada além de possibilitar seu reaproveitamento como modelo pré-treinado em diferentes cenários;
- Projetar uma *Pipeline* de *DevOps* para automatizar o ciclo de vida dos modelos pré-treinados, incluindo as etapas de criação, teste e implantação, otimizando o uso do TL;
- implementar a solução de um sistema de detecção de intrusão, usando a técnica de FTL;
- Desenvolver a arquitetura considerando as melhores praticas identificadas na revisão sistemática da literatura.

E - Avaliar o desempenho do modelo:

- Realizar os experimentos para identificar a as métricas *Accuracy* (ACC), *Area Under the Curve* (AUC), *Precision*, *Recall* e *Loss* para detecção e outros indicadores de desempenho do modelo FTL;

F - Avaliar a combinação dos modelos pré-treinados:

- Avaliar o impacto da combinação dos modelos LNN, tanto centralizados quanto descentralizados, no desempenho global do modelo proposto;
- Comparar o desempenho do modelo utilizando cada uma das versões básicas e pré-treinadas individualmente.

Esses objetivos específicos delineiam as etapas necessárias para alcançar o objetivo geral da dissertação. Cada objetivo específico contribui para a compreensão, desenvolvimento e avaliação da arquitetura proposta.

1.1 Hipótese

Neste trabalho, parte-se de duas hipóteses que serão verificadas ao longo dos estudos. A primeira hipótese sustenta que a utilização de modelos de ML baseados em LNN melhora a eficiência e a eficácia do sistema de detecção de intrusões, proporcionando um desempenho superior na identificação de anomalias e na classificação de comportamentos maliciosos.

A segunda hipótese propõe que a arquitetura desenvolvida com base na técnica de FTL descentraliza o treinamento dos modelos, aliviando o tráfego nas redes e gerando um sistema capaz de identificar um número maior de ataques, além de disseminar essas informações com maior agilidade entre diferentes organizações.

Essas hipóteses orientam a investigação e a validação dos métodos propostos, com o objetivo de demonstrar que a combinação das técnicas LNN e FTL promove avanços significativos na segurança cibernética e na detecção de intrusões em ambientes distribuídos.

2 Fundamentação Teórica

Nesta seção, são descritos os conceitos fundamentais das diversas tecnologias que norteiam o presente trabalho, tais como: Network Intrusion Detection System (NIDS), técnicas de ML, modelos de FL e TL, computação de borda, definição de conjuntos de dados, entre outras informações relevantes para a compreensão da proposta apresentada no âmbito deste trabalho de Mestrado.

Além disso, a seção apresenta a relação dos trabalhos correlatos selecionados, os quais serviram de base para a elaboração e desenvolvimento desta dissertação.

2.1 Revisão Sistemática da Literatura

A revisão sistemática da literatura tem como objetivo realizar um levantamento bibliográfico de trabalhos correlatos, de forma que a coesão conceitual obtida possa servir de referência para outros pesquisadores e alunos.

Com base nos conceitos descritos no artigo *"Guidelines for Performing Systematic Literature Reviews in Software Engineering"*, de Kitchenham e Charters (2007), foram definidos os mecanismos de busca e os critérios de seleção, visando garantir consistência e relevância nos artigos escolhidos para compor o estudo. De acordo com Biolchini et al. (2005), esse procedimento é caracterizado como uma forma estruturada de obtenção de evidências na literatura, que possam justificar a realização da pesquisa. Ressalta-se, ainda, que essa revisão deve ser conduzida a partir da definição de um protocolo de buscas e de um processo metodológico bem estabelecido.

2.1.1 Etapa 1 – Planejamento

A etapa de planejamento teve como propósito estruturar o processo de busca e seleção dos artigos. Para isso, foram definidos critérios específicos que nortearam a pesquisa bibliográfica.

A - Identificação de parâmetros de busca

Palavras chaves utilizadas para identificar artigos relacionados ao tema escolhido.

Utilizando como base o tema abordado no trabalho, foram identificados os parâmetros de busca para compor as *queries* que definem a lógica para seleção dos artigos. Para a presente pesquisa foram considerados artigos cujo os títulos ou *keywords* contivessem as palavras *"federated","transfer", "learning", "detection" e "intrusion"* além de limitar o intervalo de tempo da publicação, contido no período de 2019 à 2024.

Figura 1 – Etapas utilizadas para realizar Revisão Sistemática da Literatura

Fonte:Elaborado pelo autor

O diagrama ilustra as três etapas que estruturam o processo de Revisão Sistemática da Literatura, detalhando o planejamento, a execução e a mineração dos dados.

B - Definição de *queries* e *subqueries* utilizadas

A estrutura de pesquisa foi composta a partir da definição das interseções entre as palavras-chave e das seções dos artigos onde a busca seria realizada. As *queries* foram elaboradas considerando a combinação dos termos definidos e aplicadas às seções de título, resumo e palavras-chave dos artigos indexados.

Com o objetivo de refinar e otimizar o processo de seleção dos trabalhos relevantes, foram criadas três *subqueries*, conforme descrito a seguir:

- Primeira *subquerie*: *Query* executada no título ou *keywords* utilizando as palavras *"federated", "learning", "detection" e "intrusion"* limitada ao intervalo de publicação entre o ano de 2019 e o ano de 2024, conforme descrito na Figura 2.
- Segunda *subquerie*: *Query* executada no título ou *keywords* utilizando as palavras *"transfer", "learning", "detection" e "intrusion"* limitada ao intervalo de publicação entre o ano de 2019 e o ano de 2024, conforme descrito na Figura 3.
- Terceira *subquerie*: *Query* executada no título ou *keywords* utilizando as palavras: *"federated", "transfer", "learning", "detection" e "intrusion"* limitada ao intervalo de publicação entre o ano de 2019 e o ano de 2024, conforme descrito na Figura 4.

C - Definição de mecanismo de busca(*identificação das bibliotecas*).

A busca foi realizada em bases de dados amplamente reconhecidas na área da Ciência da Computação, que oferecem repositórios relevantes e atualizados:

- *IEEE Xplore Digital Library*
- *ACM Digital Library*

2.1.2 Etapa 2 – Execução

A execução consistiu na aplicação das *queries* estruturadas nas bases de dados definidas. O objetivo foi identificar e reunir estudos alinhados ao tema proposto, criando um repositório inicial de artigos a serem analisados.

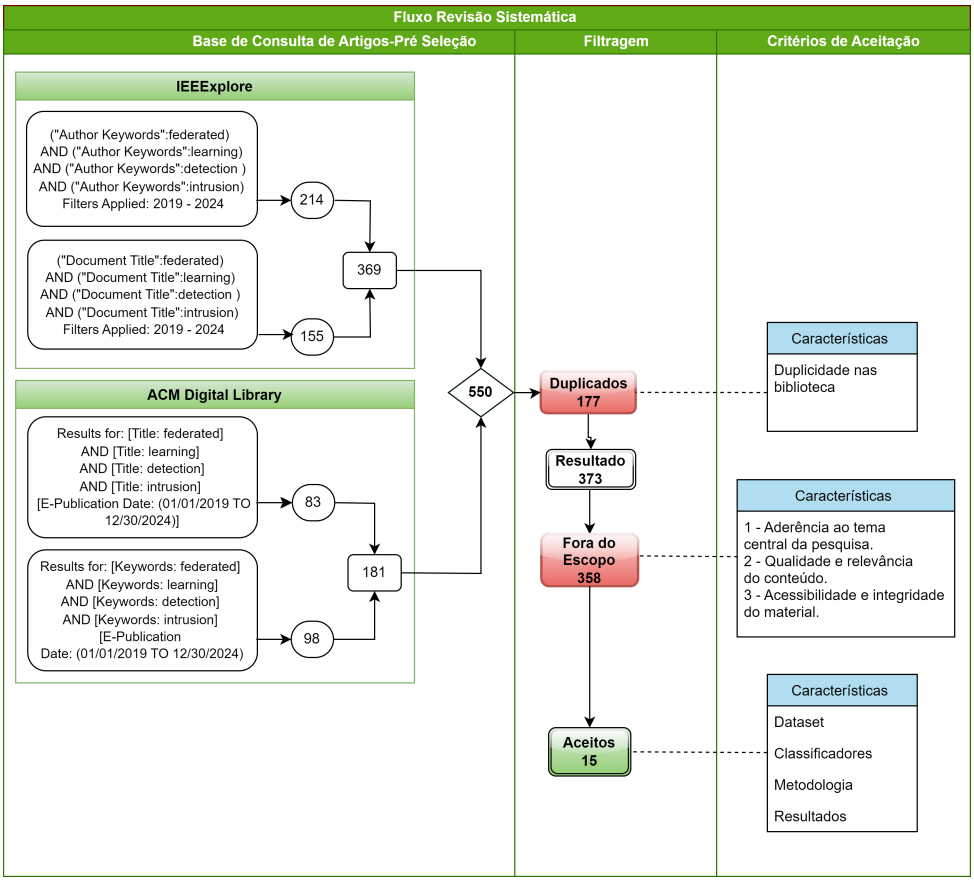
D – Levantamento de trabalhos relacionados (*Execução das queries geradas*)

Para realizar o trabalho de levantamento e execução das *queries* foram empregadas duas bases de pesquisas científicas expressivas na área da Ciência da Computação e, foram definidas as bibliotecas onde existe uma maior concentração de arquivos voltado ao tema central do trabalho: *IEEExplore* e *ACM Digital Library*.

A aplicação das *queries* resultou na coleta de 766 artigos. Esse levantamento inicial agrupou estudos em três categorias principais:

- Trabalhos sobre FL aplicados a IDS.
- Pesquisas focadas em TL em sistemas de detecção de intrusão.
- Estudos que exploram a combinação de ambas as técnicas, FTL.

Figura 2 – Fluxo Revisão Sistemática da Literatura - Federated Learning



Fonte:Elaborado pelo autor

O fluxo ilustra a aplicação da *query* nas bibliotecas *IEEE Xplore* e *ACM Digital Library*, utilizando as palavras-chave *federated*, *learning*, *detection* e *intrusion*, no período de 2019 a 2024. Na pré-seleção, foram identificados 550 artigos, dos quais 177 eram duplicados. Após a aplicação dos critérios de seleção, 358 artigos foram considerados fora do escopo, restando 15 artigos aceitos.

2.1.3 Etapa 3 – Mineração

A etapa de mineração teve como foco a filtragem e a extração de dados relevantes dos artigos obtidos na fase anterior.

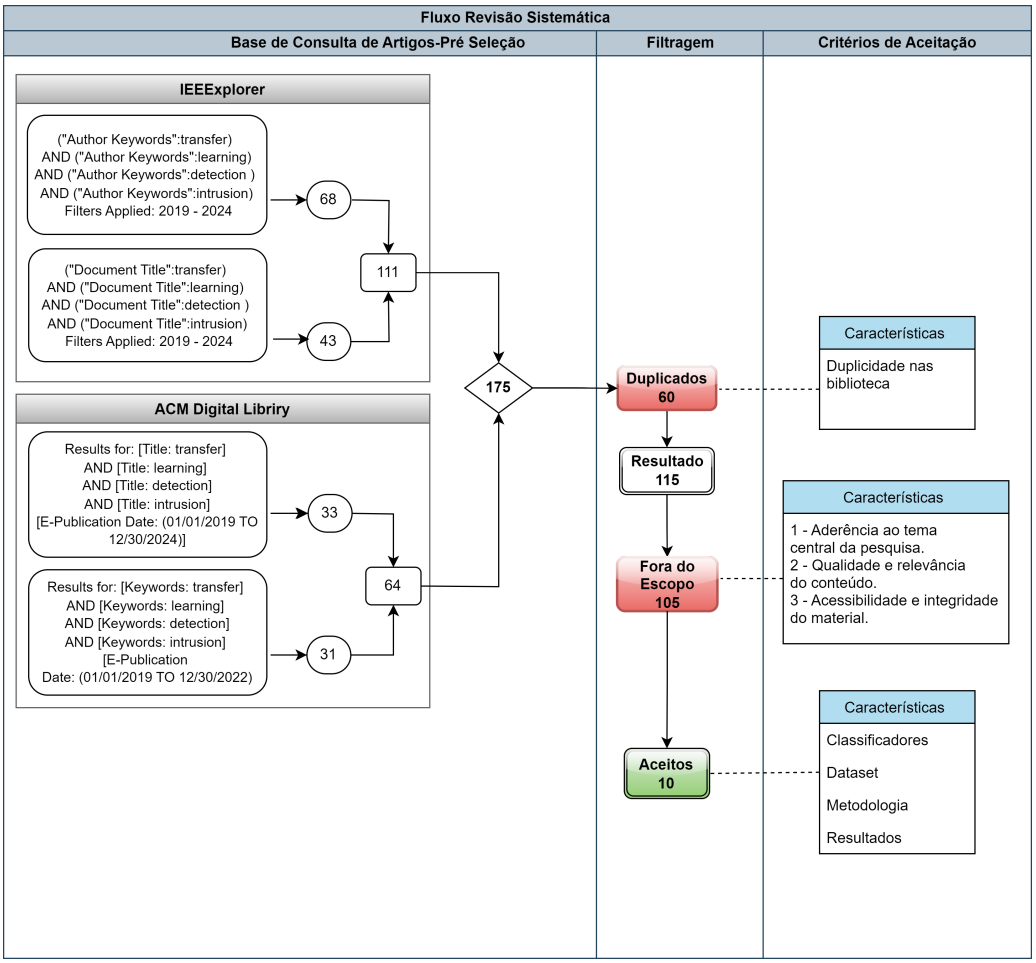
E - Filtragem de dados

A etapa consiste na aplicação dos critérios de pesquisa definidos, a partir da 1º Etapa. Em alguns casos os arquivos podem constar de forma duplicadas, inacessíveis ou até mesmo o conteúdo não estar de acordo ao trabalho proposto.

Após a execução das *queries* nas bases de dados de pesquisa mencionadas, foram recuperados 766 artigos, os quais foram inicialmente organizados em três fluxos distintos para a realização da revisão sistemática. Durante o processo, 254 artigos duplicados foram identificados e removidos, resultando em um total de 512 artigos únicos para análise.

Devido à expressiva quantidade de trabalhos ainda disponíveis após a deduplicação,

Figura 3 – Fluxo Revisão Sistemática da Literatura - Transfer Learning



Fonte:Elaborado pelo autor

O fluxo ilustra a aplicação da *query* nas bibliotecas IEEE Xplore e ACM Digital Library, utilizando as palavras-chave *transfer*, *learning*, *detection* e *intrusion*, no período de 2019 a 2024. Na pré-seleção, foram identificados 175 artigos, dos quais 60 eram duplicados. Após a aplicação dos critérios de seleção, 105 artigos foram considerados fora do escopo, restando 10 artigos aceitos.

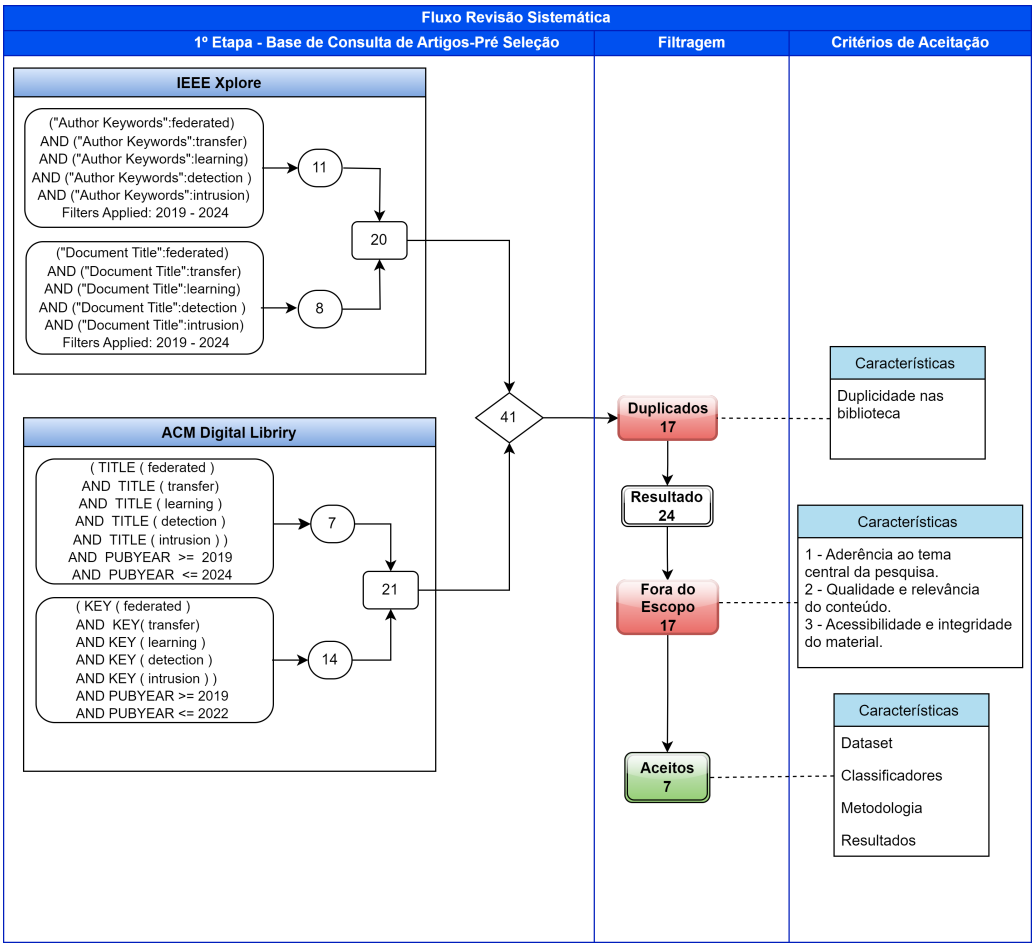
foram aplicados critérios adicionais de eliminação, considerando aspectos como a origem da publicação, título e proposta do estudo. Como resultado dessa triagem preliminar, foram selecionados 48 artigos para leitura completa.

Durante a leitura, observou-se que 17 desses artigos não apresentavam aderência ao escopo temático da pesquisa, sendo, portanto, excluídos da análise final. Ao término do processo de filtragem, 31 artigos foram considerados relevantes e alinhados ao objetivo deste trabalho, sendo então pré-selecionados para compor a base de fundamentação teórica.

A analise preliminar considerou os seguintes critérios :

- Aderência ao tema central da pesquisa.
- Qualidade e relevância do conteúdo.

Figura 4 – Fluxo Revisão Sistemática da Literatura - Federated Transfer Learning



Fonte:Elaborado pelo autor

O fluxo ilustra a aplicação da *query* nas bibliotecas IEEE Xplore e ACM Digital Library, utilizando as palavras-chave *federated*, *transfer*, *learning*, *detection* e *intrusion*, no período de 2019 a 2024. Na pré-seleção, foram identificados 41 artigos, dos quais 17 eram duplicados. Após a aplicação dos critérios de seleção, 17 artigos foram considerados fora do escopo, restando 7 artigos aceitos.

- Acessibilidade e integridade do material.

F - Extração de dados

Definição do conjunto de características considerados nos artigos para fins de análise das tendências tecnológica, relevância de conteúdo, e classificação no estado da arte do contexto do trabalho.

Etapa utilizada para selecionar conceitos significativos dos artigos e devem agregar conhecimento para realizarmos o desenvolvimento do trabalho. Foram obtidas as informações:

Os 31 artigos selecionados foram submetidos a um processo de extração de dados, visando identificar informações cruciais para a construção do referencial teórico e o delineamento do trabalho proposto. As informações extraídas incluíram:

- Técnicas de Aprendizado: Tipos de métodos utilizados nos estudos (FL, TL ou FTL);
- Classificadores Empregados: Algoritmos de aprendizado de máquina mais eficientes em cada técnica;
- Conjuntos de Dados : Bases de dados utilizadas nos experimentos;
- Métricas de Avaliação: Indicadores de desempenho como *Accuracy*, *Precision*, *Recall* e *F1-Score*;
- Cenários de Aplicação: Contextos e ambientes em que os sistemas foram avaliados.

2.1.4 Critérios de pesquisa utilizados

A seguir, são detalhados os critérios adotados durante a pesquisa bibliográfica para garantir a qualidade e a relevância dos artigos selecionados.

2.1.4.1 Parâmetros de Busca

Os critérios de busca foram definidos considerando o alinhamento dos artigos com o tema da dissertação. Foram selecionados artigos que apresentassem, no título ou nas palavras-chave, termos relacionados às técnicas estudadas e ao contexto de detecção de intrusão.

2.1.4.2 Estratégias de Filtragem

A filtragem de artigos seguiu critérios de:

- Relevância Temática: O artigo deve abordar diretamente o uso de FL, TL ou FTL em sistemas de detecção de intrusão;
- Qualidade da Publicação: Apenas publicações revisadas por pares foram consideradas;
- Atualização Temporal: O recorte temporal foi limitado aos anos de 2019 a 2024;

2.1.4.3 Ferramentas e Bases de Dados Utilizadas

A busca foi conduzida utilizando ferramentas de pesquisa avançada nas seguintes plataformas:

- *IEEE Xplore Digital Library*
- *ACM Digital Library*

Essas bases foram escolhidas por sua representatividade no campo da Ciência da Computação e por concentrarem publicações de alta relevância em aprendizado de máquina e segurança cibernética.

2.1.4.4 Resultados da Revisão Sistemática

Os 31 artigos selecionados ofereceram uma visão abrangente do estado da arte em técnicas de FL e TL aplicadas à detecção de intrusão. Os principais achados incluem:

- A predominância do uso de *Convolutional neural network* (CNN) e *Recurrent Neural Network* (RNN) em experimentos com FL;
- A aplicação recorrente de TL em cenários com escassez de dados rotulados;
- A escassez de estudos que combinam FL e TL em ambientes distribuídos, indicando uma lacuna que justifica a proposta desta dissertação.

Os dados extraídos fundamentam as decisões metodológicas adotadas neste trabalho e servem de base para a implementação e avaliação da arquitetura de FTL proposta para sistemas de detecção de intrusão.

2.2 Trabalhos correlatos

Para uma melhor visualização das informações, utilizamos uma estrutura separando os trabalhos correlatos utilizados nessa pesquisa por ano e tema.

Nesta seção são apresentados os trabalhos correlatos a partir do levantamento realizado com a parametrização previamente descrita. As subseções seguintes abordam os trabalhos por ano crescente.

2.2.1 Trabalhos correlatos - 2019

Foi identificado o artigo publicado por Nguyen et al. (2019), que aborda a utilização de FL na detecção de intrusão em dispositivos de *Internet of Things* (IoT). O trabalho expressa a preocupação dos autores com a vulnerabilidade dos dispositivos IoT, que geralmente possuem um design propenso a falhas de segurança. O ponto mais relevante do artigo é a proposta do DIoT: um sistema de autoaprendizado voltado à detecção de dispositivos comprometidos em redes IoT. Além da aplicação de FL, também são utilizadas técnicas de detecção de anomalias. Os sistemas de detecção baseados em assinaturas *(signature-based IDS)* não conseguem identificar novos tipos de ataques, ao passo que os sistemas baseados em anomalias permitem a definição de perfis de comportamento considerado normal ou esperado para cada dispositivo, facilitando a identificação de desvios.

Para implementar o algoritmo de FL, os autores utilizaram as bibliotecas *Flask* e *Flask-SocketIO* no lado do servidor, e a biblioteca *SocketIO-client* no lado do cliente. O conjunto de dados utilizado foi composto por três categorias:

- Conjunto de dados de atividade;
- Conjunto de dados de implantação;
- Conjunto de dados de ataque.

Segundo os autores, os resultados foram satisfatórios: o sistema DIoT detectou 95,6% dos ataques, com um tempo médio de resposta de 257 milissegundos e sem geração de alarmes falsos.

O artigo publicado por Wu, Guo e Buckland (2019) apresenta um estudo utilizando CNN para detecção de intrusão, alertando, entretanto, para a baixa qualidade dos conjuntos de dados disponíveis para treinamento. Conforme apontado por McHugh (2000) e Tavallaee et al. (2009), os conjuntos de dados originais DARPA (1998–1999) e KDD 1999 são altamente redundantes e contêm armadilhas que comprometem a confiabilidade na validação de modelos. Nos experimentos, foram utilizadas duas CNNs: a primeira foi treinada com um conjunto de dados base e, posteriormente, a segunda CNN foi treinada com o mesmo conjunto. Após esse processo, o modelo consolidado foi aplicado a um novo conjunto de dados com características semelhantes, caracterizando o processo de transferência de conhecimento. Os resultados demonstraram aumento de precisão em comparação aos classificadores J48, *Naive Bayes* (NB), *Naive Bayesian Tree* (NBTree), *Random Forest* (RF), *Random Tree* (RT), *Multi Layer Perceptron* (MLP) e *Support Vector Machine* (SVM).

Utilizando os indicadores Taxa de Detecção (DR), ACC e Taxa de Falsa Predição (FPR), os resultados obtidos com o conjunto de dados NSL-KDD foram os seguintes:

- KDDTest+: DR = 93,86% ACC = 87,30% FPR = 21,38%
- KDDTest-21: DR = 99,82% ACC = 81,94% FPR = 98,65%

O estudo de Zhang e Yan (2019) aborda a detecção de intrusão utilizando TL com aprendizado adversarial em redes inteligentes. A proposta apresenta uma nova perspectiva sobre os IDS, que tradicionalmente dependem de assinaturas para identificar ataques. Nesse novo modelo, o sistema utiliza aprendizado adversarial para tratar amostras rotuladas e não rotuladas, sendo capaz de identificar automaticamente ataques com características desconhecidas.

A arquitetura *Direct Artificial Neural Networks* (DANN) é composta por três redes neurais: um extrator de características, um classificador de domínio e um preditor de rótulos. O processo ocorre em quatro etapas:

1. Coleta de dados de origem rotulados e dados de destino não rotulados;

2. Treinamento do modelo em ambos os domínios para obtenção de um novo espaço de características, por meio de um processo adversarial;

3. Uso das novas representações de características dos dados de origem para treinar os classificadores;

4. Mapeamento dos dados de destino para o novo espaço de características, realizando a previsão dos rótulos de classe.

Na estrutura proposta, os autores definem dois hiperparâmetros principais: o número de características extraídas (Nf) e o valor do regularizador de adaptação de domínio (λ).

Foi realizada uma análise de ablação para avaliar a influência desses parâmetros. Os resultados indicaram que a precisão aumentava conforme Nf aumentava ou (λ) diminuía. O melhor desempenho foi alcançado com precisão de 83,7% quando Nf $=$ 60, e de 83,6% quando (λ) $=$ 0,5, valores escolhidos como os parâmetros ideais para os experimentos.

Já o artigo de Singla, Bertino e Verma (2019) trata da dificuldade em se obter conjuntos de dados adequados para o treinamento de modelos de detecção de intrusão. Os autores citam o NSL-KDD como um conjunto desatualizado e com grande volume de dados redundantes, optando, então, pelo uso do conjunto UNSW-NB15 para a realização dos experimentos. Outro problema identificado foi a necessidade de grandes volumes de dados para treinar modelos eficazes na detecção de *malwares* e ataques. Além disso, para cada novo tipo de ataque ou *malware*, é necessário recomeçar o treinamento do modelo do zero, o que representa um custo elevado. Nos experimentos, os autores utilizaram duas arquiteturas de *Deep Neural Networks* (DNN):

- *Small DNN*: com duas camadas ocultas;
- *Large DNN*: com cinco camadas ocultas.

Ao comparar os resultados, foi possível observar que o modelo proposto, mesmo utilizando uma quantidade reduzida de amostras para treinamento, obteve melhor precisão em relação ao modelo base.

Na tabela abaixo organizamos os principais detalhes extraídos dos artigos referentes ao ano de 2019:

Tabela 1 – Detalhamento dos artigos obtidos por meio da Revisão Sistemática - ano de 2019.

ID	**Artigo**	***Autor***	***Dataset***	***Classificadores***	***Método***
01	DIoT: A Federated Self-learning Anomaly Detection System for IoT	Nguyen et al. (2019)	Dados de atividade, Dados de implantação e Dados de ataque		FL

02	A Transfer Learning Approach for Network Intrusion Detection	Wu, Guo e Buckland (2019)	DARPA (1998-1999), KDD 1999 e NSL-KDD	ConvNet (Convolution Network)	TL
03	Domain-Adversarial Transfer Learning for Robust Intrusion Detection in the Smart Grid	Zhang e Yan (2019)	Cyber Attack Datasets - Industrial Control System (ICS)	*AdaBoost* (AdB), (kNN), (SVM), (RF), Classification and Regression Tree (CART), and Artificial Neural Network (ANN)	TL
04	Overcoming the Lack of Labeled Data Training Intrusion Detection Models Using Transfer Learning	Singla, Bertino e Verma (2019)	UNSW-NB15 e NSL-KDD	DNN	TL

Fonte: Elaborado pelo autor.

2.2.2 Trabalhos correlatos - 2020

As publicações relacionadas ao uso de FL para detecção de intrusão começaram a ganhar ritmo de crescimento a partir de 2020. Para esse ano, foram identificadas oito publicações relevantes.

No trabalho de Sun, Ochiai e Esaki (2020), da Universidade de Tóquio, os autores propõem uma abordagem de FL para detecção de intrusão em LANs múltiplas em larga escala. Ao definir uma quantidade específica de participantes, a proposta não consistia em treinar um único modelo e aprimorá-lo, mas sim em criar modelos específicos para cada participante e melhorá-los com base em seus próprios dados. Foram selecionados 20 participantes ao longo de 60 dias. Ao final do experimento, foram obtidas precisões médias de validação para todos os participantes com três métodos distintos de rotulagem baseados em conhecimento, alcançando resultados de 0,923, 0,813 e 0,877, respectivamente. A detecção de *malware* na rede foi realizada utilizando informações de diversos protocolos, como ARP, IP, TCP, UDP, HTTP, HTTPS, DNS, DHCP, entre outros.

Os autores Chen et al. (2020) discutem as fragilidades das *Wireless Edge Networks* (WENs) no que diz respeito à segurança, propondo o modelo *Gated Attention Gated Recurrent*

Unit (FedAGRU), baseado em FL. Vale destacar que as WENs transferem parte das funções de rede e processamento de dados da nuvem para a borda da rede, próxima ao usuário final. Os autores iniciam seus experimentos propondo uma combinação entre GRU (*Gated Recurrent Unit*) e SVM, sendo o GRU uma estrutura mais simples, com menos parâmetros e maior velocidade de treinamento.

O modelo FedAGRU possui um mecanismo de agregação no qual diferentes clientes recebem pesos distintos. O servidor mescla os parâmetros dos modelos locais de forma ponderada, acelerando a convergência e simplificando o volume de cálculos, controlados por três parâmetros principais:

A - proporção de clientes que participam de cada rodada de atualização;

B - O tamanho do minilote local usado nas atualizações dos clientes;

C - número de épocas de treinamento local por cliente.

Embora o modelo apresente uma proposta promissora, os autores utilizam o conjunto de dados NSL-KDD para os experimentos, o qual é considerado defasado e propenso a vieses, comprometendo a confiabilidade dos resultados.

No artigo escrito por Moustafa et al. (2020) tem uma proposta muito interessante. Eles abordam a utilização dos conjuntos de dados atuais mais utilizados nos experimentos e apresentam uma nova proposta, os conjuntos de dados chamados de *ToN_IoT*, que utilizam fontes de dados federadas coletadas de conjuntos de dados de telemetria de serviços IoT, conjuntos de dados de sistemas operacionais *Windows* e *Linux*, e conjuntos de dados de tráfego de rede.

O artigo apresenta um *testbed* (também conhecido como *test bed*, uma plataforma para conduzir testes rigorosos, transparentes e replicáveis de teorias científicas, ferramentas de computação e novas tecnologias) e a descrição dos conjuntos de dados *ToN_IoT* para sistemas operacionais *Windows*. Os testes foram realizados utilizando duas versões específicas do sistema operacional, *Windows 7* e *Windows 10*. As características foram extraídas dos rastreamentos de auditoria de memória, processador, processos e disco rígido.

Neste caso, o *testbed* foi implantado em três níveis: borda, névoa e nuvem. A camada de borda inclui IoT e dispositivos de rede; a camada de névoa envolve máquinas virtuais e *gateways*; e a camada de nuvem compreende serviços de nuvem. A interação dinâmica entre as três camadas foi implementada usando as plataformas *VMware NSX* e *vCloud NFV*.

Uma abordagem diferente, porém muito interessante, foi identificada no artigo elaborado pelos autores Li et al. (2020). A abordagem trouxe a utilização de métodos de Aprendizagem Federada em um Sistema de Detecção de Intrusão de Rede Distribuída em *Satellite-Terrestrial Integrated Networks* (STIN). Além das dificuldades de integrar duas redes heterogêneas, a largura de banda e a dificuldade para atualização do *hardware* após o lançamento

dos satélites são outros pontos a considerar. Com relação à privacidade dos dados trafegados, é utilizado o método de chaves para criptografar/descriptografar e autenticar dados durante a transmissão. Mesmo assim, o autor menciona a necessidade de utilizar um NIDS, que é usado para identificar o tráfego na rede, especialmente para distinguir tráfego normal e malicioso, sendo, portanto, benéfico na eliminação de tráfego malicioso. Porém, atualmente os NIDS são utilizados para redes terrestres e são difíceis de aplicar às comunicações por satélite devido às características do STIN.

Um ponto a destacar foi a arquitetura definida e a montagem do ambiente do testbed. Foram criados *hosts* utilizando *Kernel-based Virtual Machine* (KVM) e *OpenStack*, tecnologia de virtualização de rede, com servidores de alto desempenho. Outro ponto importante foi a definição do conjunto de dados, utilizando as ferramentas Argus (que processa pacotes de rede brutos) e *CICFlowMeter* (que realiza a extração de recursos baseada em fluxo). Foi gerado um arquivo e persistido na base de dados. Neste sentido, os testes foram realizados tentando cobrir nove tipos de ataques de rede terrestre (*Botnets*, *Web Attack*, *Backdoor* e seis ataques *DDoS* — LDAP, MSSQL, *NetBIOS*, Portmap, Syn, UDP) e dois tipos de ataques *DDoS* (Syn, UDP) de rede de satélite.

O artigo publicado no ano de 2020 pelos autores Xu et al. (2020) propõe um modelo baseado em CNN e TL para solucionar problemas de detecção de intrusão com reconhecimento de imagem, utilizando o conjunto de dados KDD CUP 99 para a realização de testes e experimentos. O artigo descreve todos os passos até a obtenção dos resultados. O modelo proposto utiliza a CNN, que possui grande capacidade de aprendizagem em diferentes níveis e em um grande volume de dados, no campo da detecção de intrusão. Para obter melhores resultados, foi realizada a conversão de dados de rede em imagens em escala de cinza, pois a CNN tem melhor capacidade de processamento de imagem.

Já o artigo publicado pelos autores Dhillon e Haque (2020) apresenta uma abordagem voltada para os IDS de detecção de redes (NIDS) e propõe uma metodologia utilizando o modelo híbrido *Long Short-Term Memory architecture fitted with Convolutional Neural Network in its hidden layers* (CNN-LSTM).

Inicialmente, foram realizados experimentos utilizando três algoritmos: DNN, CNN e CNN-LSTM. Dentre eles, o que apresentou melhor desempenho em detecção foi o CNN-LSTM, com ACC de 98,9%.

O conjunto de dados utilizado nos estudos foi o USNW-15, criado no *Cyber Range Lab* do *Australian Centre for Cyber Security* (ACCS), que contém nove tipos de ataques maliciosos. Sua infraestrutura de rede moderna contribui significativamente para a construção de um IDS.

A configuração utilizada para realizar os experimentos, em termos de hardware, foi um *cluster* de VM na *Google Cloud Platform* (GCP), utilizando a instância *n1-standard-16*, com 16 vCPUs e 30 GB de RAM. Para as bibliotecas de aprendizado profundo, os autores utilizaram

o *Keras Framework* com *TensorFlow* 1.15 no *backend*. O pré-processamento e a manipulação de dados foram realizados com as bibliotecas Pandas e *Scikit-learn*. Os experimentos foram executados usando o *Jupyter Notebook IDE* com *Python* 3.7.

Os resultados foram comparados com três classificadores: CNN, que apresentou uma *Precision* de 92,16%; DNN, com *Precision* de 87,66%; e CNN-LSTM, com *Precision* de 98,30%, superando os demais modelos testados.

Seguindo uma linha de pesquisa semelhante, os autores Masum e Shahriar (2020) propuseram a estrutura TL-NID (*Transfer Learning for Network Intrusion Detection*) para a detecção de intrusão em redes. Essa abordagem faz uso de uma *Deep Convolutional Neural Network* (DCNN), baseada na arquitetura *VGG-16*, previamente treinada com o conjunto de dados ImageNet.

A arquitetura da *VGG-16* é composta, principalmente, por três tipos de camadas: convolucionais, de *pooling* e totalmente conectadas. Sua estrutura é organizada da seguinte forma: inicialmente, duas camadas convolucionais são seguidas por uma camada de *pooling*; em seguida, mais duas camadas convolucionais são aplicadas, seguidas por outra camada de *pooling*, encerrando o primeiro ciclo. O segundo ciclo é composto por três blocos, cada um contendo três camadas convolucionais seguidas por uma camada de *pooling*. Por fim, a arquitetura é concluída com três camadas totalmente conectadas.

Utilizou-se a conversão das amostras de intrusão para imagens em escala de cinza, e, posteriormente, essas imagens foram transformadas em formato RGB para serem alimentadas na rede neural profunda, já que as CNN têm melhor desempenho com dados em formato de imagem.

Para os experimentos, também foram utilizados modelos convencionais como SVM, *Decision Tree* (DT), RF e *Logistic Regression* (LR), cujos resultados foram comparados com os obtidos pela estrutura TL-NID proposta. Os melhores resultados com as métricas de avaliação dos modelos nos conjuntos de dados KDDTest+ e KDDTest-21 não foram unânimes em todos os indicadores. No conjunto KDDTest+, o TL-NID alcançou os maiores valores para ACC 89,30%, *F1-Score* 90,26% e *Recall* 87,19%, enquanto os indicadores *Precision* 95,44% e taxa de falso alarme 4,41% ficaram com o DT. Já no KDDTest-21, os resultados foram diferentes, mas o TL-NID manteve-se com os melhores valores nos mesmos três indicadores: ACC 90,26%, *F1-Score* 82,48% e *Recall* 82,15%. O SVM obteve os melhores resultados para *Precision* 97,63% e taxa de falso alarme 5,1%.

O artigo publicado pelos autores Fan et al. (2020) foi o primeiro que identificamos abordando o tema FTL. Além disso, ele trata de dois assuntos em ascensão no universo tecnológico: a detecção de intrusão em redes 5G IoT. A proposta do trabalho é a utilização do *Framework IoTDefender* para a detecção de intrusão em redes 5G IoT, com base na técnica de FT. Sabe-se que as redes 5G para IoT apresentam diversos desafios a serem enfrentados,

especialmente na área de segurança. Entre eles, destacam-se: a heterogeneidade, diversidade e personalização das redes IoT; a existência de dados em vários setores no formato de ilhas isoladas; e a segregação de dados, que limita a quantidade de informações disponíveis para o treinamento de modelos eficazes de detecção de intrusão.

A rede 5G oferece várias vantagens em comparação com outras tecnologias disponíveis no mercado. Entre os principais benefícios estão: ampla cobertura, alta conectividade, baixa latência e maior segurança, aspectos que favorecem o desenvolvimento da IoT. Os serviços *Ultra-Reliable and Low Latency Communications* (uRLLC) e *Massive Machine Type Communications* (mMTC), utilizados pelo 5G, estão diretamente associados às redes IoT. O uRLLC é voltado para aplicações especiais como *Internet of Vehicles*, sistemas de controle industrial, telemedicina, entre outros. Já o mMTC permite a integração de múltiplos setores verticais, como cidades inteligentes, casas inteligentes e monitoramento ambiental.

Outra tecnologia importante no contexto do 5G é o *Mobile Edge Computing* (MEC), que realiza o processamento de dados na borda da rede, evitando o envio à nuvem. Essa abordagem contribui para reduzir a latência e aumentar a confiabilidade, sendo um grande aliado em arquiteturas baseadas em 5G e IoT.

Os IDS voltados para redes 5G IoT precisam ser modelos personalizados, pois as características do 5G — geograficamente distribuído, heterogêneo e hierárquico — não permitem a utilização de um modelo único capaz de atender a todos os tipos de ataques. Outro aspecto relevante é que as redes 5G IoT não produzem dados suficientes para o treinamento de modelos robustos, e a baixa variedade de ataques observados dificulta a generalização e prevenção de novas ameaças. Segundo os autores, um bom IDS para 5G IoT deve ser capaz de integrar dados oriundos de outras redes IoT, ampliando assim sua capacidade geral de detecção e defesa.

O modelo de aprendizado por transferência federada permite que diferentes empresas treinem seus próprios modelos personalizados e, sem compartilhá-los diretamente, adquiram conhecimento de outros modelos que compartilhem a mesma estrutura. O *IoTDefender* tem como objetivo estabelecer uma estrutura colaborativa para aprimorar o desempenho do modelo MIoTDef.

O *Framework IoTDefender* é composto por três camadas: a camada superior, chamada de *Security Cloud*, operada pela prestadora de serviços 5G, que dispõe de grande volume de dados e poder computacional para treinar o modelo do servidor; a camada inferior, composta por dispositivos IoT com diversos endpoints inteligentes; e a camada intermediária, formada pelas plataformas MEC, que hospedam o componente IDS e atuam como *gateways* de acesso local ao *Security Cloud*.

Para a realização dos experimentos, foram utilizadas quatro redes diferentes, combinadas aos conjuntos de dados CICIDS2017 (público), NSL-KDD e três conjuntos de dados IoT

(privados). Avaliou-se, então, a capacidade do *IoTDefender* em detectar ataques. Os estudos indicaram que, ao identificar pacotes maliciosos, o *IoTDefender* foi capaz de manter uma média de *Precision* de 91,93%.

Na Tabela são organizados os principais detalhes extraídos dos artigos referentes ao ano de 2020:

Tabela 2 – Detalhamento dos artigos obtidos por meio da Revisão Sistemática - ano de 2020.

ID	**Artigo**	***Autor***	***Datasets***	***Classificadores***	***Método***
05	Intrusion Detection with Segmented Federated Learning for Large-Scale Multiple LANs	Sun, Ochiai e Esaki (2020)			FL
06	Intrusion Detection for Wireless Edge Networks Based on Federated Learning	Chen et al. (2020)	KDD CUP 99, CICIDS2017 e WSN-DS	FedAGR Unidade Recorrente Gated Attention Gated Baseada em Aprendizado Federado	FL
07	Federated TON_IoT Windows Datasets for Evaluating AI Based Security Applications	Moustafa et al. (2020)	TON_IoT Windows		FL
08	Distributed Network Intrusion Detection System in Satellite-Terrestrial Integrated Networks Using Federated Learning	Li et al. (2020)	Argus e CICFlowMeter		FL

09	Intrusion Detection Based on Fusing Deep Neural Networks and Transfer Learning Communications in Computer and Information Science	Xu et al. (2020)	KDD CUP 99	SCNN	TL
10	Towards Network Traffic Monitoring Using Deep Transfer Learning	Dhillon e Haque (2020)	USNW-15	CNN-LSTM, DNN e CNN.	TL
11	TL-NID Deep Neural Network with Transfer Learning for Network Intrusion Detection	Masum e Shahriar (2020)	NSL-KDD	TL-NID - Transfer Learning for Network Intrusion Detection, SVM , LR , RF e DT.	TL
12	IoTDefender A Federated Transfer Learning Intrusion Detection Framework for 5G IoT	Fan et al. (2020)	CICIDS2017, IoT conjunto de dados - redes domésticas inteligentes, IoT conjunto de dados - rede de vigilância por vídeo de câmeras IP e NSL-KDD	IoTDefender	FTL

Fonte: Elaborado pelo autor

2.2.3 Trabalhos correlatos - 2021

Os autores Popoola et al. (2021) publicaram um artigo com um estudo desenvolvido sobre detecção de intrusão em redes heterogêneas de forma colaborativa. A proposta apresentada é a criação do algoritmo de fusão Fed+ para detecção de intrusão baseada em *Federated Deep Learning* (FDL) em redes heterogêneas.

Para a realização dos experimentos, foram utilizados quatro conjuntos de dados distintos de detecção de intrusão em redes (NF-ToN IoT-v2, NF-UNSW-NB15-v2, NF-BOT-IoT-v2 e NF-CSE CIC-IDS2018-v2), coletados de quatro redes sem fio. O modelo proposto, DNN Fed+, é composto por *Deep Learning* (DL) local na borda e pela agregação dos parâmetros dos modelos DNN locais na nuvem.

A arquitetura proposta, DNN Fed+, é composta por dois modelos: o primeiro é um modelo DL local na borda; o segundo é um modelo DNN armazenado no servidor na nuvem, responsável por realizar a agregação dos parâmetros. O DL local é implementado em nós de borda com capacidade de armazenamento e processamento de dados eficiente.

A arquitetura do modelo é composta por uma camada de entrada, duas camadas ocultas densamente conectadas e uma camada de saída. A agregação dos parâmetros do modelo no servidor na nuvem é realizada utilizando o algoritmo Fed+ e o modelo DNN local, com o objetivo de minimizar as perdas do modelo FDL.

Durante os experimentos, foi observado que os modelos locais DNN apresentaram baixa capacidade de generalização, sendo considerados inadequados para a detecção de intrusão em redes sem fio heterogêneas. Os quatro modelos FDL desenvolvidos de forma colaborativa, utilizando os conjuntos de dados mencionados e os algoritmos de fusão *FedAvg*, CM, *FedAvg+* e CM+, demonstraram desempenho de classificação mais satisfatório. Contudo, os modelos *DNN-FedAvg+* e DNN-CM+ apresentaram os melhores resultados.

O modelo *DNN-FedAvg+* alcançou os seguintes índices: ACC de 99,27 $\pm$ 0,79%, *Precision* de 97,03 $\pm$ 4,22%, *Recall* de 98,06 $\pm$ 1,72%, *F1-Score* de 97,50 $\pm$ 2,55% e FPR de 2,40 $\pm$ 2,47%. O modelo DNN-CM+ obteve: ACC de 99,28 $\pm$ 0,79%, *Precision* de 97,15 $\pm$ 3,97%, *Recall* de 98,08 $\pm$ 1,78%, *F1-Score* de 97,57 $\pm$ 2,45% e FPR de 2,27 $\pm$ 2,50%. Após a conclusão dos experimentos, foi identificado que os algoritmos *DNN-FedAvg+* e DNN-CM+ obtiveram valores mais altos de ACC, *Precision*, *Recall* e *F1-Score*, bem como uma taxa de falsos positivos (FPR) mais baixa, demonstrando que possuem uma capacidade de generalização superior em relação aos demais algoritmos utilizados durante os testes.

O artigo publicado pelos autores Mirzaee et al. (2021), propõe uma arquitetura de detecção de intrusão utilizando a técnica de *Federated Deep Neural Network* (FDNN) para monitorar o tráfego e projetar um IDS com o objetivo de garantir a segurança e a privacidade da infraestrutura de medição na *Smart Grid* (SG). No FIDS, os usuários são posicionados na *Home Area Network* (HAN) e contribuem com concentradores na *Neighbor Area Network* (NAN) para a construção de um modelo de detecção abrangente. O FL requer uma infraestrutura de comunicação entre usuários e concessionárias para o compartilhamento periódico das atualizações do modelo. Neste caso, os autores propõem que todo o processo ocorra em uma rede 5G, em razão de sua conectividade massiva, confiabilidade e eficiência.

Os experimentos realizados com outros classificadores como comparação apontaram índices bastante relevantes na utilização do FDNN. Mesmo utilizando o conjunto de dados NSL-KDD, considerado obsoleto, os autores apresentaram os seguintes resultados: 99,5% de ACC, 99,5% de *Precision* e 99,5% de *F1-Score*. Os classificadores utilizados para comparação com o FDNN foram: *Centralized DNN* (CDNN), *Support Vector Machine* (CSVM), *K-Nearest Neighborhood* (CKNN) e *Logistic Regression* (CLR).

O artigo publicado pelos autores Lalouani e Younis (2021) apresentou uma proposta diferente para os IDS utilizados na detecção de intrusão em redes IoT. Enquanto os IDS atuais se baseiam nos conceitos de NIDS, o artigo propõe uma arquitetura na qual os IDS seriam armazenados nos dispositivos finais, utilizando o conceito de *Host Intrusion Detection System* (HIDS). Esses IDS seriam implantados nos equipamentos de borda, denominados *Edge of Things* (EoT), que compõem a arquitetura da solução proposta.

A proteção de uma rede *EoT* contra ataques cibernéticos é fundamental, uma vez que esses dispositivos podem compartilhar *logs* e alertas de segurança para atualizar os recursos do IDS.

A ideia proposta pelos autores é que o processo seja iterativo e contínuo, permitindo que os equipamentos de borda transmitam os modelos mais recentes para as camadas superiores, os quais, por sua vez, são redistribuídos para os demais dispositivos. O projeto FLIDS, baseado em FL, também visa impedir que ataques de envenenamento de dados, realizados por invasores agindo individualmente ou em grupo, venham a ocorrer. Além disso, busca-se aproveitar os benefícios da metodologia FL para lidar com questões de escalabilidade e privacidade. Os autores também mensuraram, durante os experimentos, o objetivo de reduzir o tráfego de dados na rede, uma vez que os dados coletados pelos dispositivos IoT são processados parcial ou totalmente na borda, sendo em seguida compartilhados e/ou agregados nos nós em nuvem.

Para avaliar a eficácia e a eficiência do FLIDS, foi utilizado o conjunto de dados *NF-UNSW-NB15-v2*. Durante os experimentos, o FLIDS foi comparado a IDS centralizados. O desempenho foi avaliado utilizando *Neural Networks*, bem como os classificadores *AdaBoost* e *Naive Bayes*.

O FLIDS superou o *AdaBoost* e o *Naive Bayes* em termos de ACC, *Precision*, *Recall* e *F1-Score*. Os resultados demonstraram a eficácia da abordagem federada e das redes neurais. Enquanto o *AdaBoost* obteve os seguintes indicadores: *ACC* 0,9937, *Precision* 0,45, *Recall* 0,82 e *F1-Score* 0,58, o *Naive Bayes* alcançou *ACC* 0,993, *Precision* 0,44, *Recall* 0,85 e *F1-Score* 0,58. Já o FLIDS apresentou *ACC* 0,997, *Precision* 0,73, *Recall* 0,70 e *F1-Score* 0,72.

O próximo artigo aborda o NILM (*Non-intrusive Load Monitoring*), uma tecnologia que monitora e mede o consumo de energia de cada aparelho acessando sinais de energia agregados das leituras do medidor de rede. Os autores Shi et al. (2021) realizaram experimentos

utilizando FL para investigar os possíveis problemas de vazamento de privacidade do usuário em aplicativos NILM, utilizando estruturas baseadas em FL.

O alto custo de coleta e a natureza geográfica dos dados tornam raros os conjuntos de dados adequados para ambientes de aplicativos NILM. Outro ponto importante a ser ressaltado é que, devido a restrições legais ou regulamentos corporativos, os detentores de dados não podem compartilhá-los com terceiros, resultando em subconjuntos de dados isolados. Essas barreiras limitam a possibilidade de integrar dados de múltiplas fontes e treinar um modelo conjunto.

Os autores analisaram o impacto do vazamento de gradientes dos dados de treinamento na privacidade do usuário, o que pode ser entendido como informações que não estariam disponíveis durante o uso prático do modelo, mas que são acidentalmente incluídas no processo de treinamento ou validação, gerando potenciais erros a serem identificados. Essa análise foi realizada ao lidar com diversos cenários NILM dentro de uma estrutura de *FL*, utilizando o modelo CNN.

Para os experimentos, foi utilizado o conjunto de dados *UK-DALE* e o modelo CNN de última geração, com um paradigma de aprendizado *sequence-to-point* (*seq2point*).

Segundo os autores, o estudo mostrou que o uso de uma estrutura de aprendizagem federada baseada em redes neurais para aplicativos NILM pode expor dados de treinamento e comprometer a privacidade do usuário. Os dados vazados são suficientes para especular sobre a rotina diária dos moradores. Essa análise levou à conclusão de que é necessário construir uma nova estrutura de aprendizagem federada, mais segura, para garantir maior proteção à privacidade dos usuários em aplicativos NILM.

O artigo publicado pela autora Otoum e Nayak (2021) aborda o tema da detecção de intrusão em sistemas de veículos inteligentes. Segundo a autora, os *Intelligent Connected Vehicles* (ICV) utilizam uma tecnologia promissora que substituirá os veículos tradicionais em breve. O estudo visa identificar as vulnerabilidades que podem existir nos sistemas utilizados em veículos inteligentes e propõe um IDS independente de infraestrutura para proteger tanto os veículos internamente quanto as redes externas. O IDS proposto utiliza os algoritmos *Deep Belief Network* (DBN) e RF para detectar ataques nas comunicações intra e interveiculares. Além disso, nesse modelo, o IDS fica armazenado nos veículos conectados.

Os dois principais componentes de um *ICV* são: a *Controller Area Network* (CAN), que é uma rede central de topologia de barramento dentro dos veículos e conecta diferentes *ECUs* e outros dispositivos de entrada/saída, e a *Electronic Control Unit* (ECU), unidade de computação central do veículo, que controla sistemas e subsistemas conectados, como sistemas de freio, sistema de freio antitravamento (ABS), *airbags*, entre outros.

As diferentes formas de comunicação utilizadas na tecnologia *ICV* expõem os veículos a diversos tipos de ciberataques ou ameaças, tais como *Denial of Service* (DoS), *Distributed Denial of Service* (*DDoS*), ataques de *spoofing*, *sniffing*, entre outros.

O fluxo do modelo para realizar a detecção inicia com o tráfego, que é monitorado por um mecanismo de detecção baseado em assinatura. Caso alguma anomalia seja identificada, um alerta é acionado para o motorista e para o administrador do sistema de gerenciamento de segurança. Se nenhum ataque for detectado, o tráfego será classificado por meio do mecanismo de detecção utilizando o classificador DBN. No caso de um novo ataque, o log será enviado para um sistema de gerenciamento de segurança baseado em nuvem, que contém um gerador de assinaturas de ataque e um banco de dados com informações completas sobre as assinaturas e a lista de assinaturas de cada veículo conectado. Para novas assinaturas de ataque, o sistema enviará uma atualização *over-the-air* (OTA) com a nova assinatura para os veículos conectados.

Para a realização dos experimentos, foram utilizados os conjuntos de dados conhecidos CICIDS2017 e NSL-KDD. O CICIDS2017 contém tráfego normal e diversos tipos de ataques, como ataque de força bruta, ataque *Heartbleed*, *botnet*, ataque DoS, ataque *DDoS*, ataque na *web* e ataque de infiltração.

Para a seleção de atributos, foi utilizado o algoritmo *RF*. Métodos de seleção que combinam técnicas de filtragem e agrupamento foram empregados para aprimorar a escolha das características mais relevantes. A seleção de atributos auxilia na redução do *overfitting* do modelo, aumenta a velocidade de treinamento, reduz a complexidade e melhora o desempenho do modelo.

Os atributos selecionados na fase anterior são utilizados como entrada para o mecanismo de detecção de intrusão. Nesta etapa, a autora utilizou TL para reaproveitar o conhecimento adquirido (parâmetros e recursos) de um domínio de origem para um domínio de destino.

Para os experimentos, foram consideradas métricas clássicas de avaliação para técnicas de classificação, como ACC, *Precision*, taxa de detecção (*Detection Rate* - DR), *F1-Score* e a curva *Receiver Operating Characteristic* (ROC), comparando o modelo proposto com quatro classificadores distintos de ML e DL: RNN, DNN, SVM e *Adaptive Boosting* (*AdaBoost*).

Os resultados demonstraram que o modelo proposto superou os demais modelos tradicionais de ML/DL utilizados no mesmo ambiente, como RNN, DNN, SVM e *AdaBoost*, alcançando uma *Precision* de 93,95% e 94,51% nos conjuntos de dados NSL-KDD e CICIDS2017, respectivamente.

Tabela 3 – Detalhamento dos artigos obtidos por meio da Revisão Sistemática - 2021.

ID	Artigo	*Autor*	*Dataset*	*Classificadores*	*Método*
13	Federated deep learning for collaborative intrusion detection in heterogeneous networks.	Popoola et al. (2021)	(NF-ToN IoT-v2, NF-UNSW-NB15-v2, NF-BOT-IoT-v2 e NF CSE CIC-IDS2018-v2		FL
14	FIDS A Federated Intrusion Detection System for 5G Smart Metering Network	Mirzaee et al. (2021)	NSL-KDD	FDNN - Federated Deep Neural Network,	FL
15	Robust distributed intrusion detection system for edge of things	Lalouani e Younis (2021)	NF-UNSW-NB15-v2	Adaboost, Naives Bayes e FLIDS	FL
16	Signature-Over-The-Air com Transfer Learning IDS para Intelligent Connected Vehicles	Otoum e Nayak (2021)	CIDS2017 e NSL-KDD	RNN, DDNN, SVM e Adaptive Boosting (AdaBoost)	FL
17	User Privacy Leakages from Federated Learning in NILM Applications	Shi et al. (2021)	UK-DALE	CNN (seq2point)	TL

Fonte: Elaborado pelo autor.

2.2.4 Trabalhos Correlatos - 2022

No estudo realizado por Bertoli, Júnior e Saotome (2022), os autores abordam a detecção de ataques de varredura em redes heterogêneas, destacando sua importância na segurança de sistemas distribuídos. A varredura, etapa inicial de muitos ataques, deve ser identificada e bloqueada precocemente para minimizar danos e otimizar recursos computacionais. No entanto, grande parte das pesquisas sobre Sistemas de Detecção de Intrusão em Rede não con-

sidera o contexto de redes IoT e os desafios da arquitetura *Zero Trust*, que exige mecanismos de defesa distribuídos em cada dispositivo.

Para validar sua abordagem, os autores utilizaram o conjunto de dados TON_IoT, que reflete a complexidade das redes atuais, abrangendo computação em nuvem e dispositivos IoT na borda e na nuvem. A versão modificada, NF-ToN-IoT-v2, foi processada com *NetFlow*, gerando 43 atributos de fluxo de rede para análise.

Os experimentos foram conduzidos com Regressão Logística utilizando o classificador SGD da biblioteca *scikit-learn*, escolhido por sua simplicidade e eficiência em testes preliminares. A implementação do FL seguiu a abordagem *FedAvg*, permitindo a agregação descentralizada dos modelos locais. Foi adotado um aprendizado federado horizontal, onde os 13 alvos da base de dados foram divididos em 13 subconjuntos, mantendo a mesma estrutura de atributos, mas com amostras distintas.

A análise de desempenho considerou a *F1-Score*, métrica mais adequada para cenários com desequilíbrio de classes. Os resultados demonstraram que o modelo global treinado com *FedAvg* superou o treinamento local tradicional em 12 dos 13 subconjuntos, evidenciando as vantagens do FL na melhoria da detecção de ameaças em redes heterogêneas.

O estudo de Aouedi et al. (2022) explora o FL semi-supervisionado para IDS, demonstrando como técnicas de ML/DL podem aprimorar a detecção de ameaças, tornando a gestão e implantação de redes mais flexíveis e automatizadas. O IDS pode ser implementado como uma função de rede virtual (VNF) e distribuído em diferentes camadas da infraestrutura.

Os autores destacam que a abordagem FL semi-supervisionada reduz custos ao dispensar a necessidade de rotulação manual dos dados nos nós de borda. Em vez disso, os dispositivos aprendem representações de dados não rotulados por meio de um Autoencoder (AE), enquanto o servidor central utiliza umapequena porção de dados rotulados para treinar um modelo supervisionado (Rede Neural).

A arquitetura proposta inclui um servidor FL na nuvem, que agrega os AE dos clientes e treina o modelo supervisionado. O conjunto de dados UNSW-NB15, composto por nove categorias de ataques, foi utilizado para os experimentos, sendo previamente normalizado para otimizar o treinamento.

Para avaliar o desempenho do modelo, foram comparadas abordagens supervisionadas e semi-supervisionadas. Nos experimentos com amostras rotuladas, o FL semi-supervisionado superou algoritmos tradicionais como DT, SVM e RF, alcançando ACC de 84,32%, *Precision* de 86,19%, *Recall* de 83,10% e *F1-Score* de 83,63%. Já na análise com dados não rotulados, o modelo proposto também obteve os melhores resultados em relação à abordagem Non-FL, confirmando sua eficácia na detecção de intrusões com menor dependência de dados rotulados.

A ideia principal do artigo é abordar a vulnerabilidade *zero day*, no contexto de detecção de intrusão, escrito pelos autores He et al. (2022) traz uma abordagem diferente para o

nosso trabalho. A maior parte dos artigos que utilizamos nesse estudo utilizam métodos de detecção baseados em *software*, esse artigo propõe técnicas de detecção de *malware* (*software* malicioso) baseada em *hardware* (HMD), segundo os autores, essa técnica surgiu como um substituto promissor para superar as ineficiências e sobrecargas de desempenho das estratégias convencionais de detecção baseadas em software.

Os métodos HMD treinam algoritmos padrão de ML em eventos HPC para desenvolver classificadores precisos para detectar assinaturas de software mal-intencionado. Os HPCs são registradores especializados construídos em microprocessadores modernos para coletar os eventos de *hardware* de aplicativos em execução Gao et al. (2021), Wang et al. (2020).

Para desenvolver o trabalho os autores Identificaram alguns desafios nos métodos de detecção baseados em HPC existentes atualmente e proporam uma solução baseada em aprendizado profundo para realizar uma detecção de *malware Zero day* baseada em *hardware*.

O *Deep-HMD*, uma abordagem baseada em DNN de dois estágios para detecção precisa e eficaz de *malware Zero day* baseada em *hardware*. Ele primeiro transforma *malware* baseado em HPC e dados benignos em imagens e, em seguida, aproveita uma abordagem leve de aprendizado profundo para obter um alto desempenho de detecção de malware (para testes conhecidos e desconhecidos), apesar de usar um pequeno número de eventos de hardware capturados na *execution-time* por HPCs existentes.

Para elaborar o *framework* os autores realizaram estudos a procura dos melhores recursos. Uma das escolhas foi a *ResNet* , rede neural convolucional que implementa blocos residuais de "pular conexões" para aliviar o problema de desaparecimento do gradiente, configurando um atalho alternativo para a passagem do gradiente. Além disso, eles permitem que o modelo aprenda uma função de identidade. Isso garante que as camadas superiores do modelo não tenham um desempenho pior do que as camadas inferiores.

Para modelar o tipo de ameaça de *malware Zero day* os autores utilizaram nove tipos de *malware*, considerando todos os quatro tipos de *malware* de *rootkit*, *backdoor*, vírus e *ransomware* como os dados de teste de *Zero day* de destino. Esses quatro tipos de *malware* não são apresentados nos conjuntos de dados de treinamento e teste conhecidos, portanto, o conjunto de *malware* de *Zero day* é totalmente desconhecido do conjunto de dados de treinamento, os outros cinco tipos de *malware*, incluindo *trojan*, *spyware*, *botnet*, *worm* e *adware*, bem como o restante das amostras benignas, são considerados para fins de treinamento e testes conhecidos.

Os resultados de desempenho do *Deep-HMD* versus diferentes detectores baseados em ML para detecção de *malware zero day* usando 4 eventos HPC. Foram escolhidos pelos autores detectores baseados em ML que são amplamente adotados nas técnicas HMD existentes.

Foram treinados classificadores de aprendizado de máquina mais robustos sobre os dados tabulares e aplicamos a rede *Deep-HMD* nos dados de imagem incorporados bidimen-

sionais gerados a partir do mesmo conjunto de dados tabulares que foram alimentados pelos classificadores ML padrão. O *Deep-HMD* proposto atinge 97% na medida F1, 97% na área sob a Curva (AUC) e 98% de *Precision* para detecção de *malware* desconhecido. Além disso, obtém 96'% de *Precision* e 99% de *Recall*. Considerando que, o classificador de ML padrão de melhor desempenho, RF, pode atingir apenas 79% em F-measure, 87% em AUC e *Precision* e 68% em *Recall* quando usado para detectar *malware* desconhecido.

Visando ataques que podem ser realizados em sistemas disponíveis em veículos conectados a Internet de Veículos (IoV), os autores Yang e Shami (2022) realizam sua pesquisa e propõe um modelo de IDS inteligente baseado em redes neurais convolucionais otimizadas (CNNs), aprendizado de transferência e técnicas de aprendizado de conjunto. Cinco modelos avançados de CNN, incluindo *VGG-16*, *VGG-19*, *Xception*, *Inception* e *InceptionResnet* Petrov e Hospedales (2019), são utilizados para treinar os conjuntos de dados de tráfego de rede de veículos. A eficácia e a eficiência da estrutura IDS proposta são avaliadas usando dois conjuntos de dados de rede pública de veículos: o conjunto de dados *Car-Hacking* Seo, Song e Kim (2018), que representa dados intra-veículo, gerados pela transmissão de pacotes CAN no barramento CAN de um veículo real, o conjunto de dados *Car-Hacking* envolve quatro tipos principais de ataques: *DoS*, *fuzzy*, *spoofing* de engrenagem e ataques de spoofing de rotações por minuto (RPM). Já o conjunto de dados CICIDS2017 representa dados de rede externa, pois é um conjunto de dados de segurança de rede de última geração que inclui os padrões de ataque mais atualizados, eles podem ser resumidos em cinco tipos: ataques DoS, ataques de varredura de porta, ataques de força bruta, ataques da *web* e *botnets*.

O modelo utilizado no IDS proposto é composto pelos modelos básicos CNN (*VGG-16*, *VGG-19*, *Xception*, *Inception* e *InceptionResnet*) que irá realizar um *Ensemble* para gerar um modelo único.

Os modelos CNN são otimizados pelo *Particle Swarm Optimization* (PSO), um método *Hyperparameter Optimization* (HPO) que pode ajustar automaticamente os hiperparâmetros. Depois, os 3 modelos CNN de melhor desempenho são selecionados como os três modelos CNN básicos para construir os modelos de aprendizado conjunto. Por fim, duas estratégias de conjunto, média de confiança e concatenação, são usadas para construir modelos de conjunto para detecção final conforme descrito pelos autores, o objetivo do trabalho é desenvolver um IDS capaz de detectar vários tipos de ataques em redes veiculares internas e externas para proteger ambas.

Para desenvolver o IDS proposto, tanto para IVNs, quanto para redes veiculares externas, dois conjuntos de dados são usados neste trabalho. Os resultados obtidos nos experimentos apontam que a estrutura IDS proposta pode efetivamente identificar vários tipos de ataques, a avaliação dos modelos CNN otimizados e dos modelos de conjunto propostos nos conjuntos de dados *Car-Hacking* e CICIDS2017, foram obtidas pontuações F1 mais altas de 100% e 99,925% respectivamente.

O artigo publicado pelos autores Pawlicki, Kozik e Choraś (2022) avalia o uso de técnicas de aprendizado de transferência para reduzir os efeitos da mudança de implantação na detecção de intrusão de rede baseada em aprendizado de máquina.

Para realizar os experimentos os autores utilizaram a biblioteca de código aberto e de alto desempenho *TensorFlow*, junto com *Keras*, que por sua vez é uma biblioteca de rede neural de código aberto escrita em *Python*. Durante os experimentos também foi utilizado o *scikit-learn* Pedregosa et al. (2011) que também é uma biblioteca de aprendizado de máquina de código aberto para a linguagem de programação *Python*.

Para realizar os experimentos, foram utilizados dois conjuntos de dados de *benchmark* baseados em fluxo de rede foram usados. NF-CSE CIC-IDS2018, baseado no CICIDS2018 Sharafaldin, Lashkari e Ghorbani (2018) e NF-BOT-IoT, baseado em BOT-IoT dataset Komisarek et al. (2021).

Segundo os autores, o experimento inovador apresentado no artigo, tinha como objetivo descobrir a quantidade mínima de dados necessária para ajustar uma rede pré-treinada para manter o desempenho adequado em uma nova rede, porém em análises, com treinamento e conjuntos de testes provenientes da mesma distribuição, produz resultados precisos, porém o modelo obtido por este procedimento não funciona tão bem na mesma tarefa, usando os mesmos recursos, mas em dados coletados em uma rede diferente. Os experimentos sugerem que apenas 100 amostras de cada classe são suficientes para atualizar o componente em uma nova situação, porém os autores deixam a entender que a pesquisa não foi suficiente.

O artigo de Aprendizado por Transferência Federado, escrito por Wang et al. (2022) retrata a criação de um *Framework FLTrELM* baseado em aprendizagem por transferência federada e aprendizado extremo, tem como proposta resolver os problemas de ilhas de dados, escassez de amostras rotuladas, proteção de privacidade de dados e personalização na detecção de intrusão. O *framework* utiliza aprendizado federado Yang et al. (2019) e criptografia homomórfica Zhang et al. (2021) para construir um poderoso modelo de máquina de aprendizado extremo, agregando dados de instituições independentes enquanto protege a privacidade dos dados. Depois que o modelo é construído, o FLTrELM reutiliza o método de aprendizado por transferência para obter um modelo ou algoritmo personalizado para cada organização, dessa forma, resolve a falta de amostras rotuladas e distribuição de dados incompatíveis e melhora efetivamente o efeito de detecção de intrusão.

Segundo os autores, os resultados dos experimentos mostram que o FLTrELM, em comparação com os métodos tradicionais de aprendizado de máquina, melhora substancialmente a precisão da detecção de intrusão, especialmente para menos amostras e novas invasões. Adicionalmente, ao mesmo tempo, através da transferência de conhecimento, obtém-se um forte modelo de aprendizagem que inclui comportamentos individualizados aplicáveis a várias instituições.

Para verificar a eficácia e o desempenho de generalização do algoritmo proposto de detecção de intrusão FLTrELM, três conjuntos de dados de detecção de intrusão foram utilizados : NSLKDD, KDD99 e ISCX2012, sendo que o conjunto de dados ISCX2012 possui as amostras com características mais próximas ao que se espera diagnosticar nos experimentos, testes e treinamento.

O artigo publicado pelos autores Cheng et al. (2022) aborda o tema de detecção de intrusão utilizando as técnicas de FTL em equipamentos de bordas, ou podemos chamar de MEC. Outra técnica abordada no artigo é o aprendizado por reforço (RL), que é utilizada para selecionar clientes a fim de não prejudicar o treinamento do modelo. A proposta de utilizar o FTL é alcançar a preservação da privacidade e o treinamento do modelo de comunicação eficiente.

O *framework* FTL proposto para detecção de intrusão em equipamentos de borda é composto por duas etapas principais. Na primeira etapa, um modelo bem treinado é selecionado no domínio de origem e transferido para o servidor de borda no domínio de destino, onde é utilizado para o treinamento do modelo FL. Portanto, o primeiro problema é projetar uma estrutura FTL eficiente para o sistema considerado. O segundo problema é projetar um esquema de seleção de cliente baseado em RL para FTL.

O modelo proposto pelos autores demonstra o primeiro passo sendo a seleção do modelo inicial com as melhores características, na sequência o modelo é transferido para o servidor, utilizando técnicas de TL, após essa etapa é introduzido o modelo de RL para realizar a seleção de clientes. Nos equipamentos MEC é realizado o treinamento dos modelos que após a execução dessa etapa, envia os dados treinados utilizando técnicas de FL, deve ser observado que todas as técnicas proposta no artigo são utilizada como se fosse uma engrenagem.

Durante os experimentos foram utilizados dois conjuntos de dados amplamente usados em pesquisas de detecção de intrusão, NSL-KDD e UNSW-NB15 para avaliar o desempenho. Neste caso, NSL-KDD é usado no domínio de origem e UNSW-NB15 é usado no domínio de destino.

Os resultados dos experimentos demonstram melhores índices quando comparados ao FTL sem seleção do cliente com o FL. Para essa avaliação foi assumido que não há clientes maliciosos e identificou-se que o FTL tem maior precisão inicial e maior velocidade de aprendizado do que o FL, tanto para a validação quanto para o teste de curvas de precisão. Em termos de desempenho de convergência, o FTL também é superior ao FL, especialmente na precisão da validação. Para a precisão da validação, o FL atinge apenas 75% enquanto o FTL atinge 85% . Para a precisão do teste, o FL atinge apenas 70% enquanto o FTL atinge 75% . Um ponto importante a considerar nesse estudo é a consequência que pode ser gerada com a utilização de amostras ou dados gerados pelos clientes de baixa qualidade, o desvio que ele pode dar no treinamento realizado.

Na abordagem apresentada por Ji, Zhang e Ma (2022) os autores trazem a utilização de um novo método de detecção de intrusão FTLCNN, que integra aprendizado de transferência federada e rede neural convolucional com o intuito de resolver os problemas de escassez de amostra, desequilíbrio e adaptação de probabilidade sob o mecanismo de aprendizado federado. O FTLCNN reúne dados de detecção de intrusão de diferentes organizações por meio de aprendizado federado e criptografia homomórfica, de modo a estabelecer um modelo confiável de aprendizado de máquina e proteger a privacidade dos usuários. Para realizar os experimentos os autores utilizam uma estrutura onde representa quatro organizações e um servidor sem perder a generalidade.

A estrutura é composta principalmente de três partes: primeiro, treinar o modelo global que está armazenado no servidor na nuvem utilizando como base o conjunto de dados públicos; então, o modelo global é distribuído para todas as organizações, e cada organização usa o modelo global e os próprios dados para treinar o modelo global, nesta etapa, devido à grande diferença de distribuição entre o servidor e os dados da organização, foi realizado o aprendizado por transferência para gerar um modelo mais adequado para organização; por fim, os modelos de transferência criados pelas organizações serão transferidos para o servidor central para ajudar geração de um novo modelo global, armazenado na nuvem, por meio da agregação realizada entre os modelos será gerado um modelo final de aprendizado forte após muitas iterações.

O algoritmo utilizado no FTLCNN foi implementado utilizando a linguagem *Python*. A CNN é usada para treinar e prever a rede no servidor e no cliente. O modelo proposto é composto por três camadas de convolução, três camadas de *pooling* e duas camadas de conexão completa, com núcleo de convolução. Para realizar a etapa de treinamento foi utilizado o conjunto de dados de detecção de intrusão UNSW-NB15, desenvolvido pelo laboratório *Australian Network Security Center* (ACCS) em 2015.

Para identificar se o FTLCNN obteriam bons índices durantes os experimentos, ele foi submetido a uma comparação utilizando os seguintes classificadores RF-GBDT, CNN, *AdaBoost* e KNN para verificar as vantagens do algoritmo proposto. O desempenho obtido pelo FTLCNN frente aos outros algoritmos foi satisfatório, os resultados experimentais finais mostram que o modelo FTLCNN tem maior taxa de detecção, menor taxa de falso positivo e taxa de falso negativo. A taxa de detecção é de 86,85% a taxa de falso positivo é de 1,54% e a taxa de falso negativo é de 1,78%

Na abordagem apresentada por Zhang et al. (2022), os autores destacam o potencial de vulnerabilidades ao comparar o hardware de rede tradicional com dispositivos IoT e IIoT, os quais, por possuírem geralmente baixos custos de fabricação e menor manutenção, tornam-se mais suscetíveis a ataques. Outro ponto relevante do trabalho é a proposta de aprimoramento do desempenho do sistema de detecção de intrusão em ambientes IIoT. Para isso, os autores introduzem um algoritmo de agregação de classificações baseado em uma abordagem de

votação ponderada. Além disso, propõem uma estratégia de reforço de agregação em duas etapas, utilizando um modelo abrangente, capaz de atender de forma eficiente todos os clientes distribuídos, com o objetivo de melhorar a qualidade da agregação do modelo.

Para avaliar a distribuição dos dados e atribuir pesos adequados aos modelos, é aplicada a métrica de *Maximum mean discrepancy* (MMD), permitindo definir pesos para modelos bem ajustados, utilizando a votação ponderada para as previsões finais. Essa estratégia visa otimizar o uso dos modelos atualizados durante a agregação, evitando a simples média de parâmetros como ocorre no método *FedAvg*. Os experimentos realizados demonstraram um desempenho superior na detecção de intrusão, alcançando 95,97% de precisão com o AdaBoost e 73,70% com o RF, superando os modelos de linha de base em até 12,72%

Tabela 4 – Detalhamento dos artigos obtidos por meio da Revisão Sistemática - ano de 2022.

ID	**Artigo**	***Autor***	***Dataset***	***Classificadores***	***Método***
18	Improving detection of scanning attacks on heterogeneous networks with Federated Learning.	Bertoli, Júnior e Saotome (2022)	TON_IoT, NetFlow e NF-ToN-IoT-v2(conjunto de dados adaptados)	FedAvg e Regressão logística	FL
19	Intrusion detection for Softwarized Networks with Semi-supervised Federated Learning.	Aouedi et al. (2022)	UNSW-NB15	DT, SVM e, RF	FL
20	Deep Neural Network and Transfer Learning for Accurate Hardware-Based Zero-Day Malware Detection	He et al. (2022)	NSL-KDD e UNSW-NB15	Deep-HMD, DNN	TL

21	A transfer learning and optimized CNN based intrusion detection system for Internet of Vehicles	Yang e Shami (2022)	Car-Hacking e CICIDS2017	CNN(*VGG-16*, *VGG-19*, Xception, Inception e InceptionResnet)	TL
22	Towards Deployment Shift Inhibition Through Transfer Learning in Network Intrusion Detection	Pawlicki, Kozik e Choraś (2022)	CICIDS2018 e BOT-IoT	DNN	TL
23	Federated Transfer Learning With Client Selection for Intrusion Detection in Mobile Edge Computing	Cheng et al. (2022)	NSL-KDD e UNSW-NB15		FTL
24	A Novel Method of Intrusion Detection Based on Federated Transfer Learning and Convolutional Neural Network	Ji, Zhang e Ma (2022)	UNSW-NB15	FTLCNN e CNN	FTL
25	An Efficient Intrusion Detection Method Based on Federated Transfer Learning and an Extreme Learning Machine with Privacy Preservation	Wang et al. (2022)	NSL-KDD, KDD99 e ISCX2012	FLTrELM, ELM e SVM	FTL

26	Federated Learning for Distributed IIoT Intrusion Detection using Transfer Approaches	Zhang et al. (2022)		(AdaBoost e RF	FTL

Fonte: Elaborado pelo autor.

2.2.5 Trabalhos Correlatos - 2023

Na análise do artigo de Karimireddy et al. (2023), os autores relacionam as principais características para identificar um bom *framework* de FL, destacam a necessidade de uma análise sistemática dos *frameworks* de FL, motivados nas tomadas de decisão de pesquisadores, identificam *frameworks* adequados e lacunas de pesquisa futuras. Desta forma, analisam 14 *frameworks* FL (*Tensorflow Federated*, *PySyft FATE*, *PaddleFL*, *FedBioMed*, *Substra*, *FedML*, *FLower*, *FederatedScope*, *OpenFL*, *NVFLARE*, *APPFL*, *Vantage6* e *FedN*) focando características como distribuição de dados, topologias de comunicação e segurança. Assim, de forma conclusiva, os autores finalizam que *FedML* e *Federated Scope* se destacam por suas funcionalidades abrangentes.

O artigo de Lazzarini, Tianfield e Charissis (2023) a proposta tem como base um método para detecção de intrusão utilizando FL. Para o desenvolvimento da pesquisa os autores utilizam ferramentas e conjunto de dados de código aberto, onde foram alcançados resultados discretos. Para validar a pesquisa utilizaram o conjunto de dados ToN_IoT e CICIDS2017, fazendo uma classificação binária e multiclasses, com um modelo de rede neural artificial (ANN) partilhado e a média federada (*FedAvg*) como algoritmo de agregação. Com objetivo de enriquecer os resultados, os autores avaliam mais três métodos de agregação alternativos, nomeadamente *FedAvgM*, *FedAdam* e *FedAdagrad*, e comparam os seus desempenhos com o *FedAvg*.

O artigo elaborado pelos autores Rajesh et al. (2023) parte de duas premissas centrais. A primeira refere-se à escassez e à dificuldade na obtenção de dados específicos para ambientes IIoT. A segunda limitação está relacionada à ineficiência dos algoritmos tradicionais de ML, que têm sido amplamente investigados para o desenvolvimento de NIDS baseados em ML aplicados ao IIoT. Devido à alta dimensionalidade dos dados provenientes de dispositivos IoT e à heterogeneidade das fontes de onde esses dados são coletados, os algoritmos convencionais tendem a perder eficiência e apresentar dificuldades de adaptação. Como solução, os autores propõem o uso do FTL para realizar a detecção de intrusão em redes IIoT, tendo como peça central uma nova arquitetura de rede neural combinacional, denominada Combo-NN, capaz de

melhorar a eficácia do processo de detecção. Destaca-se ainda a escolha do conjunto de dados utilizado, considerado abrangente, pois reúne informações coletadas de diversos dispositivos IIoT. Essa abordagem se mostra agnóstica quanto às características específicas do conjunto de dados, permitindo a utilização de qualquer dataset de IoT de código aberto para fins de análise. A arquitetura proposta combina uma DNN com uma CNN como base para a aplicação do FTL no ambiente IIoT, resultando na criação da Combo-NN. Para a condução dos experimentos, os dados de treinamento existentes, denotados por *Xtrain*, foram divididos em duas partes: os dados do cliente (Xc) e os dados do servidor (Xs).

Os resultados obtidos mostram uma pequena variação no desempenho ao longo das execuções. Especificamente, observou-se que o Cliente 2 atingiu uma precisão de 89,8%, enquanto o Cliente 1 alcançou 89,7% após a primeira iteração. O servidor, por sua vez, atingiu uma precisão de 90% nessa etapa inicial. Após a segunda iteração, o Cliente 2 manteve seu desempenho com precisão de 89,8%, enquanto o Cliente 1 apresentou uma leve redução, alcançando 88,2%. O servidor também apresentou uma ligeira queda, atingindo 86% de precisão, mas ainda assim mantendo um desempenho elevado.

Tabela 5 – Detalhamento dos artigos obtidos por meio da Revisão Sistemática - ano de 2023.

ID	**Artigo**	***Autor***	***Dataset***	***Classificadores***	***Método***
27	Federated learning showdown: The comparative analysis of federated learning frameworks.	Karimireddy et al. (2023)			FL
28	Federated learning for iot intrusion detection.	Lazzarini, Tianfield e Charissis (2023)	ToN_IoT e CICIDS2017	Rede neural artificial (ANN)	FL
29	Give and Take: Federated Transfer Learning for Industrial IoT Network Intrusion Detection	Rajesh et al. (2023)		Combo-NN, CNN e DNN	FTL

Fonte: Elaborado pelo autor.

2.2.6 Trabalhos Correlatos - 2024

No estudo de Cevallos-Salas et al. (2024), os autores realizam uma pesquisa sobre a detecção e classificação de *malware* de privacidade ofuscado, utilizando técnicas de análise de despejo de memória. Os autores desenvolvem três classificadores: um classificador binário que distingue entre amostras benignas e maliciosas usando regressão logística, e dois classificadores multiclasses que categorizam o *malware* em diferentes tipos, como *spyware*, *ransomware* e cavalos de Tróia. A pesquisa destaca a eficácia de uma nova arquitetura *Ensemble*, onde um modelo de uma DNN, combinado com um modelo de LR em comparação com algoritmos de Aprendizado de Máquina tradicionais, demonstrando melhorias estatisticamente significativas nas métricas de desempenho. Em conclusão, o estudo evidencia a importância de abordagens inovadoras na detecção de *malware*, especialmente em um cenário onde as técnicas de ofuscamento estão se tornando cada vez mais sofisticadas. A utilização de conjuntos de dados como o CIC-MalMem-2022 e a aplicação de técnicas como o *SMOTE* para lidar com desequilíbrios de dados são passos cruciais para o avanço na proteção da privacidade dos usuários contra ameaças cibernéticas.

No estudo, proposto pelos autores Chen et al. (2024) os desafios enfrentados atualmente no campo da detecção de intrusão são cuidadosamente analisados, destacando-se o potencial das técnicas de FL e TL para superá-los, especialmente no contexto da segurança de redes.

O algoritmo FETLSVM emprega o Aprendizado Federado para treinar o modelo e realizar o compartilhamento criptografado dos parâmetros, garantindo a privacidade das informações. Essa etapa do processo é dividida em três partes principais: o aprendizado do modelo na nuvem, o aprendizado do modelo em cada organização participante e a atualização do modelo central na nuvem.

Os resultados experimentais, realizados utilizando o conjunto de dados CICIDS2017, demonstram que o algoritmo proposto apresenta vantagens significativas em relação aos algoritmos de referência, comprovando sua eficácia e precisão. As métricas de ACC obtidas para diferentes modelos — *AdaBoost*, *GoogLeNet*, *AlertNet*, *DeepFed* e FETLSVM — evidenciam esse desempenho superior.

Para as amostras Benign, os valores de acurácia foram, respectivamente: 95,46%, 96,48%, 95,21%, 98,34% e 98,89%. Nas amostras Brute Force, os resultados foram 95,45%, 92,99%, 94,21%, 97,66% e 97,23%. Para as amostras de XSS, os valores alcançados foram 10,88%, 11,21%, 12,45%, 53,32% e 65,35%, enquanto para as amostras de SQL Injection, os resultados foram 0,45%, 1,27%, 2,86%, 15,57% e 22,85%, evidenciando um ganho expressivo no desempenho, especialmente com o uso do FETLSVM.

Tabela 6 – Detalhamento dos artigos obtidos por meio da Revisão Sistemática - ano de 2024.

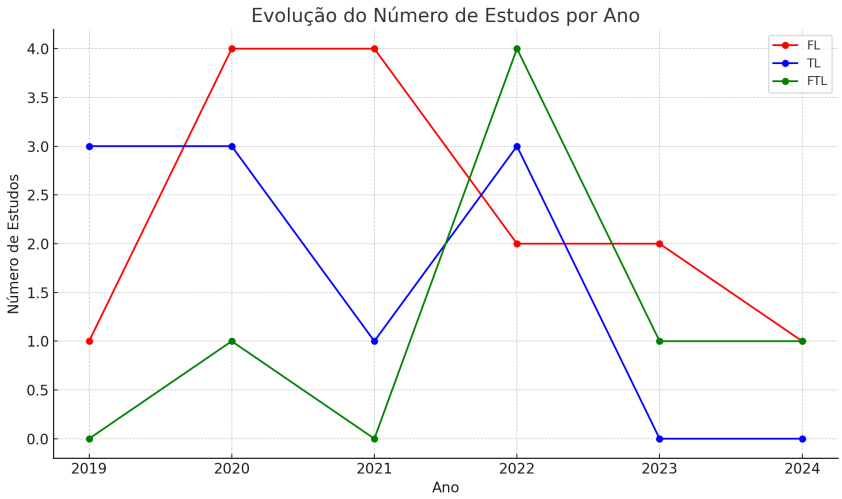
ID	Artigo	*Autor*	*Dataset*	*Classificadores*	*Método*
30	Obfuscated privacy malware classifiers based on memory dumping analysis.	Cevallos-Salas et al. (2024)	CIC-MalMem-2022	DNN e LR	FL
31	Federated learning for iot intrusion detection.	Chen et al. (2024)	CICIDS2017	AdaBoost, GoogLeNet, AlertNet, DeepFed e FETLSVM	FTL

Fonte: Elaborado pelo autor.

2.2.7 Compilação dos trabalhos utilizados Revisão Sistemática da Literatura

A Seção a seguir documenta o processo de compilação dos trabalhos utilizados na Revisão Sistemática da Literatura de forma a destacar as tendências e os resultados obtidos. A compilação dos trabalhos realizados identificou-se que o grande volume dos artigos publicados nas bibliotecas utilizadas no estudo, ainda reflete as técnicas de TL e FL. Conforme apresentado na Figura 5, o número de publicações que utilizam a abordagem de FTL ainda se mostra discreto, indicando que se trata de um campo emergente e com amplo potencial de exploração.

Figura 5 – Arquivos selecionados por Ano/Método.



Fonte: Elaborado pelo autor.

O gráfico reflete a evolução dos artigos publicados para as técnicas TL, FL e FTL no período de 2019 à 2024.

2.2.7.1 Tendências extraídas - visualização das características

Realizando uma análise quantitativa das três variáveis utilizadas na estrutura do trabalho (dataset, classificador e método) para mapear as tendências para características fundamentais dos trabalhos obtidos.

Identificamos que o comportamento adotado nas pesquisas e experimentos para as técnicas de FL, TL e FTL não seguem um padrão continuo para as variáveis utilizadas como base na estrutura do trabalho, conjunto de dados e classificador. Os classificadores básicos de ML são ajustados ou recriados com o objetivo de obter melhores indicadores (ACC, *Precision*, *Recall* e *F1-Score*), em alguns casos, existe a criação de um *Framework* levando como nomenclatura alguma característica do estudo. Já para os conjuntos de dados, além de serem utilizados os conjuntos tradicionais, também há a utilização de informações coletadas através de alguns dispositivos ligados ao estudo, por exemplo: Se estamos realizando um estudo referente a carros autônomos, podemos gerar um conjunto de dados que será utilizado para a realização dos testes e treinamentos com informações geradas por dispositivos e programas que compõe o sistema do veículo, dessa forma, identificamos que a criação de métricas referentes essas duas variáveis pode não trazer nenhum valor agregado para o nosso trabalho.

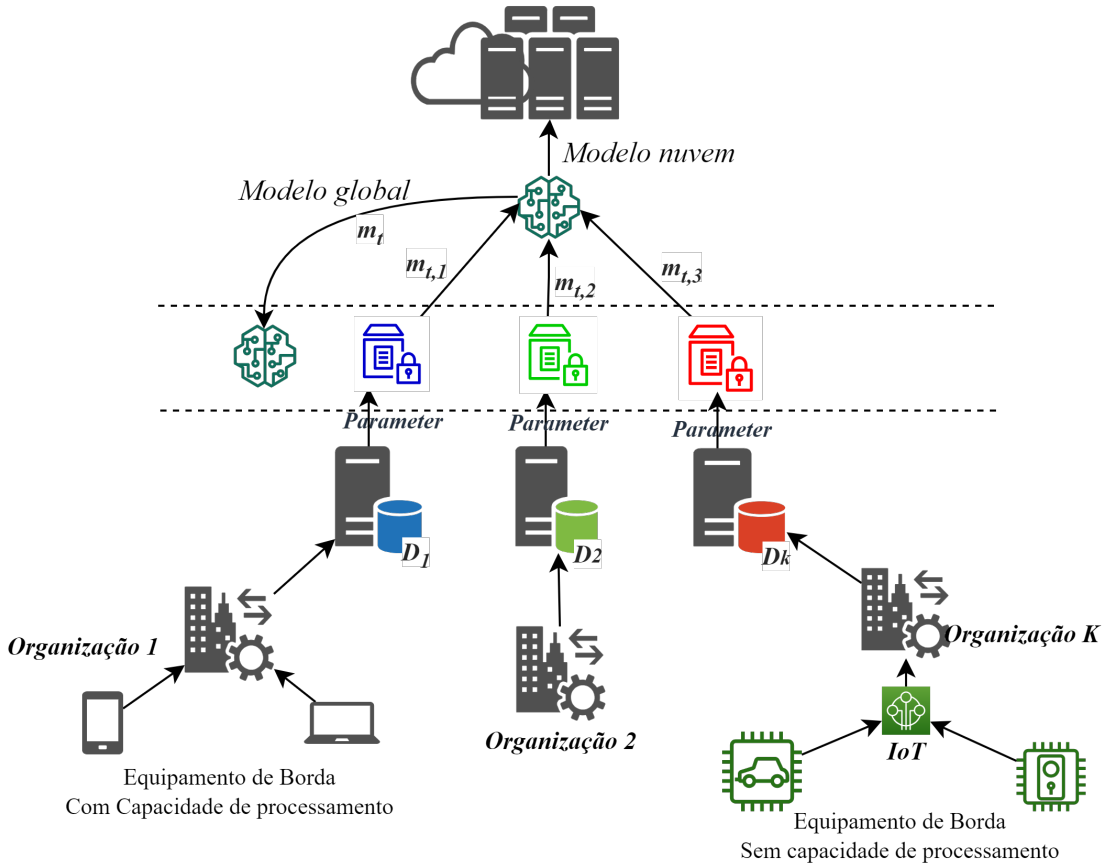
Durante a análise dos artigos que compõem a Revisão Sistemática da Literatura, observou-se a ausência de uma estrutura clara que detalhe como deve ser realizada a orquestração dos modelos quando o cliente ou a organização necessitar transferir um modelo pré-treinado para seu ambiente. Essa lacuna evidencia a necessidade de uma abordagem que viabilize a transferência de forma eficiente, segura e automatizada. Diante desse cenário, esta dissertação propõe que a orquestração do processo de transferência seja realizada por meio de uma *Pipeline* de *DevOps*, permitindo a automação das etapas de criação, teste e implantação dos modelos pré-treinados, além de garantir a rastreabilidade, escalabilidade e conformidade durante o processo de transferência.

2.3 Federated Learning (FL)

O aprendizado Federado foi utilizado pela primeira vez em um artigo publicado pelo *Google* em 2016 *Communication-Efficient Learning of Deep Networks from Decentralized Data* McMahan et al. (2016), porém alguns relatos costumam definir que o primeiro artigo publicado referente ao assunto foi em 2017 *Collaborative Machine Learning without Centralized Training Data*, escrito pelos autores Brendan McMahan e Daniel Ramage Learning (2017), desde então surgiram vários estudos sobre o assunto.

O FL é utilizado para treinar modelos de ML de forma descentralizada e colaborativa. Nessa abordagem, os dados e os modelos permanecem armazenados em dispositivos de borda, os quais são responsáveis por treinar localmente seus próprios classificadores. Em seguida, os parâmetros dos modelos (e não os dados) são transferidos para um servidor central, onde

Figura 6 – Diagrama Federated Learning



Fonte: Adaptado de Wang et al. (2022).

No cenário proposto, o Servidor Central recebe o modelo/parâmetros treinado pelos respectivos Servidores Locais, realiza um novo processamento para agregar as novas características ao modelo global, distribui o novo modelo para tarefas de execução nos Servidores Locais, transfere o novo modelo para os equipamentos de borda.

ocorre a consolidação dos resultados provenientes de diferentes dispositivos.

O modelo global atualizado é, então, retransmitido aos dispositivos de borda, permitindo a continuidade do treinamento em ciclos sucessivos. Esse processo iterativo possibilita o aprendizado colaborativo sem a necessidade de compartilhamento direto dos dados sensíveis, conforme descrito na Figura 6 e discutido por Yang et al. (2019) e Saha e Ahmad (2021). De acordo com os diferentes padrões de distribuição de amostras e espaços de características, a FL pode ser dividida em três categorias:

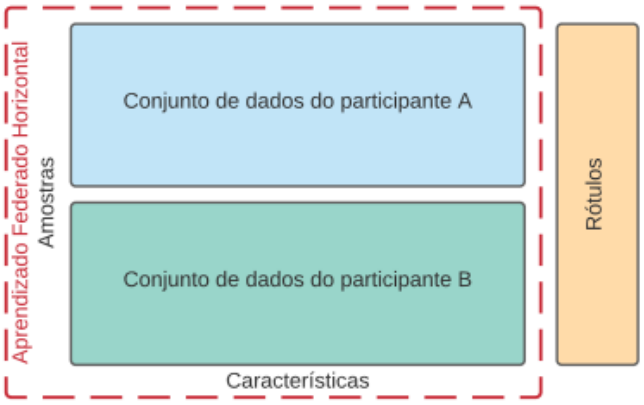
2.3.1 Horizontal Federated Learning

A aprendizagem federada horizontal, ou aprendizagem federada baseada em amostras, é introduzida nos cenários em que os conjuntos de dados compartilham o mesmo espaço de recursos, mas diferem no espaço de identificação da amostra Yang et al. (2019) (Figura 7).

Em 2017, o Google propôs uma solução de aprendizagem federada horizontal para atualizações de modelos de telefones Android McMahan et al. (2017). Nessa estrutura, um único usuário com um telefone Android atualiza os parâmetros do modelo localmente e carrega os parâmetros na nuvem Android, treinando assim o modelo centralizado junto com outros proprietários de dados.

Por exemplo, dois bancos regionais podem ter grupos de usuários muito diferentes de suas respectivas regiões, e o conjunto de interseção de seus usuários é muito pequeno. No entanto, seus negócios são muito semelhantes, portanto, os espaços de recursos são os mesmos. Shokri e Shmatikov (2015) propôs um esquema de aprendizado profundo colaborativo onde os participantes treinam independentemente e compartilham apenas subconjuntos de atualizações de parâmetros.

Figura 7 – Horizontal Federated Learning.



Fonte: Neto, Mattos e Fernandes (2020).

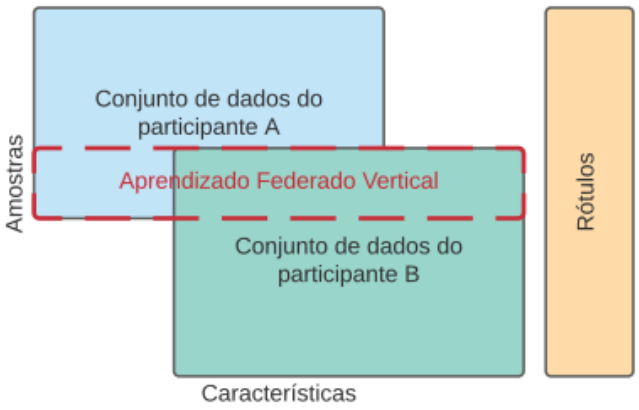
O aprendizado federado horizontal utiliza conjuntos de dados que possuem o mesmo espaço de características, mas diferem no espaço de amostras.

2.3.2 Vertical Federated Learning

O aprendizado federado vertical ou o aprendizado federado baseado em recursos (Figura 8) é aplicável aos casos em que dois conjuntos de dados compartilham o mesmo espaço de identificação de amostra, mas diferem no espaço de recursos Yang et al. (2019).

Por exemplo, considere duas empresas diferentes na mesma cidade, uma é um banco e a outra é uma empresa de comércio eletrônico. É provável que seus conjuntos de usuários contenham a maioria dos residentes da área, portanto, a interseção de seu espaço de usuário é grande. No entanto, como o banco registra o comportamento de receita e despesa do usuário e a classificação de crédito, e o comércio eletrônico retém o histórico de navegação e compra do usuário, seus espaços de recursos são muito diferentes. Suponha que queremos que ambas as partes tenham um modelo de previsão para compra de produtos com base nas informações do usuário e do produto.

Figura 8 – Vertical Federated Learning.



Fonte: Neto, Mattos e Fernandes (2020).

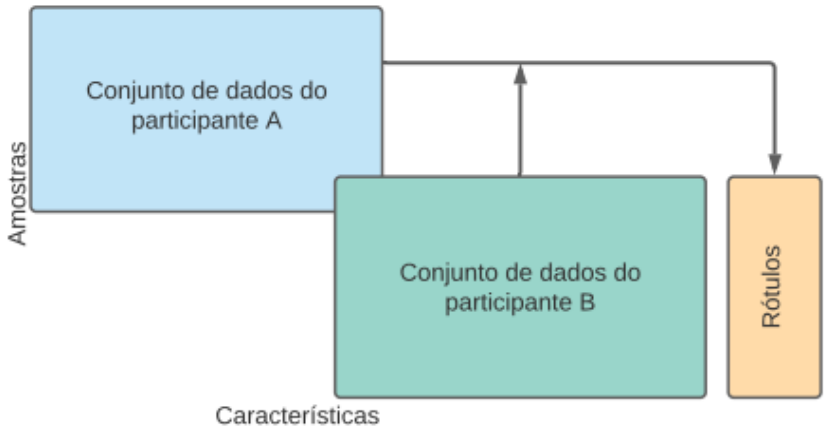
No aprendizado federado vertical, o conjunto de dados possui algumas amostras semelhantes, porém com características diferentes. Antes do início do treinamento, os participantes A e B selecionam, de uma forma segura, a interseção dos espaços de amostras.

2.3.3 Federated Transfer Learning

O Aprendizado por Transferência Federada se aplica aos cenários em que os dois conjuntos de dados diferem não apenas em amostras, mas também no espaço de recursos ou características. Este tipo de FL, Liu et al. (2020) é diferente dos dois algoritmos de FL anteriores. Neste caso, é usado quando o usuário e as características do usuário dos dois conjuntos de dados raramente se sobrepõem, sem segmentar os dados, mas usando o aprendizado de transferência Pan e Yang (2010) (a transferência de conhecimento de um campo existente para um novo campo relacionado a ele) para superar a falta de dados ou rótulos são frequentemente usados para resolver o problema de diferentes espaços de recursos de conjuntos de dados e a escassez de amostras de rótulos.

Considere duas instituições, uma é um banco localizado na China e a outra é uma empresa de comércio eletrônico localizada nos Estados Unidos. Devido a restrições geográficas, os grupos de usuários das duas instituições possuem uma pequena interseção. Por outro lado, devido aos diferentes negócios, apenas uma pequena parte do espaço de recursos de ambas se sobrepõe. Neste caso, as técnicas de aprendizado por transferência Pan e Yang (2010) podem ser aplicadas para fornecer soluções para toda a amostra e espaço de recursos sob uma federação (Figura 9). Especialmente, uma representação comum entre os dois espaços de recursos é aprendida usando os conjuntos de amostras comuns limitados e posteriormente aplicada para obter previsões para amostras com apenas recursos de um lado.

Figura 9 – Federated Transfer Learning



Fonte: Neto, Mattos e Fernandes (2020).

No aprendizado por transferência federada, o participante A utiliza todas as características do conjunto de dados do participante B para fazer sua aprendizagem. Utilizando técnicas de aprendizado por transferência, que transfere o conhecimento de um modelo para outro sem expor o conjunto de dados que foi utilizado para treinar o modelo de origem.

2.4 Transfer Learning (TL)

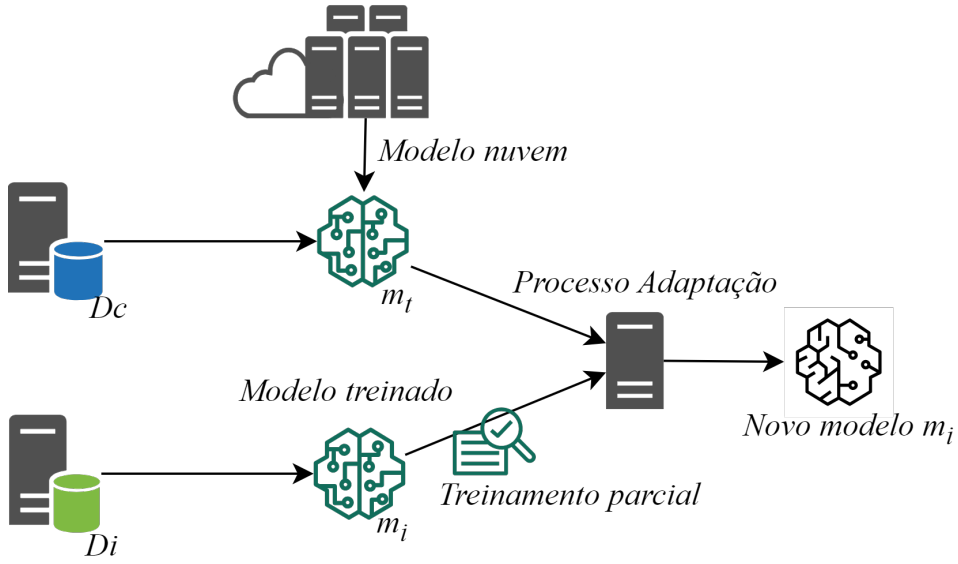
O aprendizado por transferência pode ser considerado como um dos maiores ganhos para otimizar o processo de treinamento de um determinado conjunto de dados utilizando no cenário de ML. O aprendizado por transferência é a técnica de reutilizar um modelo que foi treinado anteriormente em um determinado conjunto de dados que possua característica similares ao que deve possuir o novo conjunto de dados que será treinado pelo mesmo modelo Zhang e Yan (2019).

Os modelos de ML precisam de conjuntos de dados com um grande volume de dados para serem treinados e obter um aprendizado para trabalhar de forma eficaz. O aprendizado por transferência ajuda, nesse desafio, pois como o modelo já foi utilizado em uma etapa de treinamento anterior, o aprendizado de transferência pode trabalhar com menos dados e em um tempo mais curto, economizando muitos recursos computacionais e reduzindo o custo de construção de modelos (Figura 10). A técnica de reutilização de um modelo pré-treinado em um novo conjunto de dados vem sendo muito utilizada em grandes empresas de tecnologia Wu, Guo e Buckland (2019).

Bozinovski e Fulgosi publicaram em 1976 o artigo *The influence of pattern similarity*Bozinovski e Fulgosi (1976) *and Transfer of learning upon training of a base perceptron B2* abordando o tema de aprendizado por transferência no treinamento de redes neurais. Outra contribuição expressiva foi a de Pratt, em 1993 ele formulou formulou o algoritmo de transferência baseada em discriminabilidade (DBT) Pratt (1993). Os principais modelos pré-treinados incluem *Oxford VGG Model, Google Inception Model, Microsoft ResNet Model,*

Google word2vec Model, Stanford's GloVe Model, Caffe Model Zoo etc.

Figura 10 – Diagrama Transfer Learning.



Fonte: Adaptado de Wang et al. (2022).

O modelo global, que está no servidor na nuvem (mt), deve ser treinado através do conjunto de Dados público (Dc), o modelo pré-treinado (mt) pode ser reutilizado por outros participantes que por sua vez, pode agregar novas características ao modelo. O novo modelo pode transferir as novas características para o modelo de nuvem, essa nova aprendizagem pode ser transferida para os demais participantes.

2.5 Sistemas Detecção de Intrusão (IDS)

Os sistemas de detecção de intrusão são compostos por conjuntos de regras e configuração que buscam, através da análise de pacotes de uma rede, emitir alertas sobre possíveis situações que caracterizam um ataque, ou seja, uma situação que coloca em risco seu funcionamento e a segurança de seus dados. Essa análise é realizada geralmente através da comparação com os dados de ataques previamente conhecidos Camargo et al. (2021).

Em *Intrusion detection systems with deep learning: A systematic mapping study*, os autores Osken et al. (2019) definem a publicação do artigo "*Computer Security Threat Monitoring and Surveillance*", por Anderson (1980), o primeiro relato de um IDS. Outro fato que vale a pena ressaltar sobre o surgimento dos IDS ocorreu em 1986 e posteriormente em 1993 na publicação cientifica de modelos de detecção de intrusão baseados em análises estatísticas do tráfego de rede *An Intrusion-Detection Model* Denning (1987), ambos criados por Dorothy Denning e Peter Neumann em Denning .

Uma das abordagens mais utilizadas atualmente no contexto de IDS e a baseada em IA. Por meio dela, buscam-se métodos semelhantes ou parecidos ao processo de aprendizagem humana para utilização em análise de dados, como reconhecimento de padrões e otimização,

dentre outras atividades. Em se tratando da aplicação de métodos de ML, existem diversas abordagens, resultando em diferentes classificadores que podem ser utilizados na inferência de dados de rede. Existem dois principais tipos de IDS, o NIDS (*Network Intrusion Detection System*) e o HDIS (*Host Intrusion Detection System*). O objetivo principal de ambos é a detecção de anomalias ou alerta de eventos anormais e maliciosos que possam comprometer sua rede ou até mesmo seu computador. O nosso trabalho será pautado em cima dos conceitos de NIDS Lucas et al. (2021)

2.5.1 Network Intrusion Detection System (NIDS)

Atualmente os diversos ambientes de redes integradas, enfrentam um aumento significante de dispositivos conectados, com o volume de informação que são trafegadas entre sistemas e os dispositivos IoT que também estão conectados em redes. A internet aparece provendo suporte de conectividade para toda estrutura proposta, desde dispositivos de consumo até sistemas de controle industrial. Consequentemente, esse aumento de dispositivos conectados causa aumento do tráfego na rede e exposição dos dispositivos a agentes maliciosos. Nesse contexto, a necessidade de segurança cibernética é obrigatória para manter essas soluções operacionais sem impactar os serviços e usuários dos sistemas (Figura 11).

O funcionamento do NIDS é utilizado para análise do tráfego da rede e verificar operações suspeitas. Basicamente funcionam como *Packet Sniffers* que captura o tráfego e usam regras predefinidas ou outros parâmetros de maneira a verificar se a rede está ou não comprometida. Lucas et al. (2021)

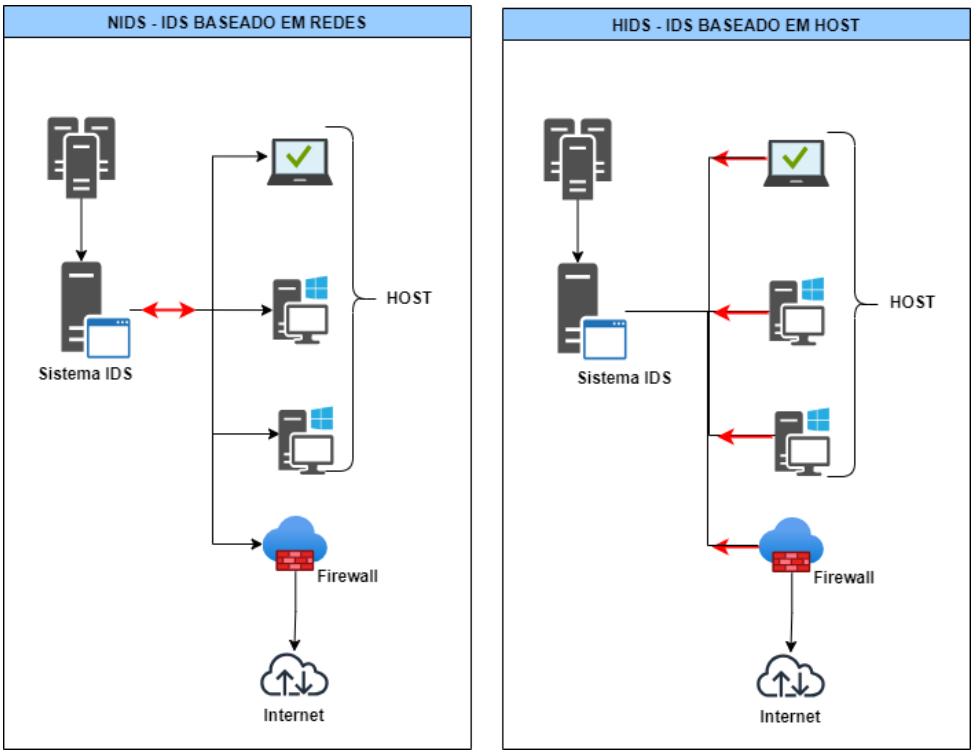
2.5.2 Host Intrusion Detection System (HDIS)

Enquanto o objetivo do NIDS é monitorar e identificar ataques realizados a uma rede, o HIDS foca na prevenção e detecção de ataques contra um computador ou computadores. Além disso, o HIDS também pode diagnosticar falhas de segurança ou vulnerabilidades em equipamentos, ele dispõe de diversas ferramentas voltada para a gestão de políticas, análise de estatísticas e outras ferramentas importantes para um bom e eficaz funcionamento do IDS (Figura 11). Os recursos disponíveis em uma ferramenta HIDS possibilita que a infraestrutura de uma rede possa ter uma proteção maior. Lucas et al. (2021)

2.6 Conjuntos de dados

Os Conjuntos de dados são coleções estruturadas de informações utilizados nos treinamentos e testes para validar experimentos nos estudos de ML máquina e outros algoritmos de processamento de dados, para nosso trabalho exploramos os Conjuntos de dados utilizados em Sistemas de Detecção de Intrusão, que contêm amostras de ataques (benígnos) e outros provenientes de ataques (maliciosos).

Figura 11 – Diagrama NIDS x HIDS.



Fonte: Adaptado de Deb et al. (2019).

Os diagramas ilustram a diferença entre um IDS NIDS (que protege a rede ao monitorar o tráfego de dados) e um IDS HIDS (que protege o host ao monitorar arquivos, processos internos e atividades locais).

Existem tipos de conjuntos de dados disponíveis para diferentes domínios, como visão computacional, processamento de linguagem natural, ciência de dados, entre outros. Para nosso trabalho, utilizamos os conjuntos de dados públicos de detecção de intrusão.

A avaliação das técnicas de detecção de intrusão depende da disponibilidade de conjuntos de dados de intrusão. Em geral, a maioria das pesquisas de detecção de intrusão de rede ainda é baseada em conjuntos de dados simulados, devido à escassa disponibilidade de dados de redes reais, porém alguns estudos já apresentam experimentos utilizando conjuntos de dados obtidos em ambientes produtivos. Vale ressaltar que, conjuntos de dados reais não são facilmente acessíveis devido a questões de privacidade, uma boa forma para avaliar e comparar a qualidade de diferentes sistemas de detecção de intrusão de rede é a utilização de conjunto de dados públicos, ao longo dos anos diversos conjuntos de dados públicos de detecção de intrusão foram desenvolvidos e disponibilizados na comunidade de segurança para realização de testes e estudos Catillo et al. (2022).

A qualidade e a usabilidade dos conjuntos de dados refletem sua adequação para auxiliar os pesquisadores no desenvolvimento de técnicas avançadas de detecção. No estado da arte, inúmeras soluções disponíveis na literatura capitalizam conjuntos de dados de detecção de intrusão bem conhecidos, alcançando altos níveis de ACC e *recall*. Atualmente temos diversos

conjuntos de dados utilizados nos experimentos para detecção de intrusão foram produzidos e compartilhados para serem explorados de forma geral e outros possuem características especificas que atendem apenas experimentos específicos.

O primeiro conjunto de dados público de detecção de intrusão desenvolvido pela DARPA (*Defence Advanced Research Project Agency*) em 1999, fornecendo um conjunto de dados de *benchmarking* de detecção de intrusão abrangente e realista, denominado KDD-CUP'991 . Este é um conjunto de dados muito citado em artigos e também muito utilizado para realizar os experimentos de estudo na detecção de intrusão. Ele contém dados de fluxo rotulados resumidos com uma ampla seleção de invasões e instâncias livres de ataques simuladas em uma rede militar. O conjunto de dados abrange sete semanas de tráfego de rede e consiste em cerca de 5 milhões de linhas. Mesmo que o KDD-CUP'99 já tenha duas décadas, ele ainda é utilizado no campo da detecção de intrusão e ML, porém já identificamos em muitos artigos críticas referentes a sua estrutura e redundância de informações. O conjunto de dados NSL-KDD Tavallaee et al. (2009) foi construído a partir do conjunto de dados KDD CUP'99, com a exclusão de amostras duplicadas e redução de tamanho, também tem sofridos duras críticas pelos pesquisadores, podemos considerar que ambos conjuntos de dados estão defasados para o mundo das pesquisas de M atuais Catillo et al. (2022).

Outro conjunto de dados de detecção de intrusão público mais recente é o UNSW-NB153 Moustafa e Slay (2015). O conjunto de dados com origem no *Australian Centre for Cyber Security* (ACCS), foi criado em 2015 por meio da ferramenta *IXIA Perfect Storm*, utilizada como gerador de tráfego normal e anormal. Inclui nove categorias dos tipos de ataque modernos e tráfego normal.

Um conjunto de dados de detecção de intrusão público muito utilizado nos experimentos em artigos publicados na comunidade acadêmica e com boa popularidade entre os pesquisadores de segurança é certamente CICIDS20179 Sharafaldin, Lashkari e Ghorbani (2018), desenvolvido pelo *Canadian Institute for Cybersecurity* (CIC) em 2017 e simula dados de rede do mundo real.

Tabela 7 – Resumo comparativo de conjuntos de dados públicos para detecção de intrusão

Dataset	**Year**	***Dataset features***	***Defense Modules***	***Size***
KDD-CUP'99	1998	Other	No	5M points
NSL-KDD	1998	Other	No	150k points
UNSW-NB15	2015	Packets, other	No	2M points
CICIDS2017	2017	Packets, bidir. fows	No	3.1M fows

Fonte: Catillo et al. (2022).

2.7 Edge computing

Um tópico importante a considerar nesse trabalho é a computação de borda, diversas arquiteturas de FTL utiliza os computadores de borda para realizar o treinamento do conjunto de dados, em alguns casos há a proposta da instalação de um IDS nesse equipamento, onde além da capacidade de processamento seria explorado outros recursos.

A computação de borda é aquela na qual o processamento acontece no local físico (ou próximo) do usuário ou da fonte de dados. Ao colocar serviços de computação mais perto desses locais, os usuários aproveitam serviços mais rápidos e estáveis com uma experiência melhor. Já as empresas se beneficiam podendo disponibilizar melhor suporte a aplicações sensíveis à latência, identificar tendências e oferecer soluções e serviços aprimorados.

A computação de borda é diferente da computação em nuvem tradicional. É um novo paradigma de computação que executa a computação na borda da rede Cao et al. (2020). Sua ideia central é tornar a computação mais próxima da fonte dos dados Satyanarayanan (2017). Os pesquisadores têm diferentes definições de computação de ponta. Shi et ai. Shi et al. (2019) - Shi et al. (2016) introduziu o surgimento do conceito de *edge computing*: *Edge computing* é um novo modo de computação de execução de borda de rede. Neste momento, a computação de borda é necessária para compartilhar a pressão da nuvem e se encarregar das tarefas dentro de seu escopo de borda.

2.8 Deep Learning

O aprendizado profundo revolucionou o campo da inteligência artificial, permitindo avanços notáveis em vários domínios, incluindo visão computacional (Chauhan et al. 2024a), *Natural Language Processing* (NLP) (Devlin et al. 2019), inferência causal (Chauhan et al. 2023a) e aprendizado por reforço (Li 2017). As redes neurais profundas (DNNs) padrão provaram ser ferramentas poderosas para aprender representações complexas a partir de dados. No entanto, apesar de seu sucesso, as DNNs padrão permanecem restritivas em determinadas condições Chauhan et al. (2024).

Um dos principais benefícios do aprendizado profundo é a capacidade de lidar com *big data*. À medida que o volume de dados aumenta, as técnicas tradicionais de ML podem se tornar ineficientes em termos de desempenho e precisão. A aprendizagem profunda, por outro lado, continua a ter um bom desempenho, tornando-a uma escolha ideal para aplicações com muitos dados.

Destacamos para o nosso trabalho as MLPs*(multi-layer perceptrons*), tipo de DL utilizam uma arquitetura densa e totalmente conectada, permitindo que cada neurónio de entrada se ligue a cada neurónio de saída. Esta arquitetura permite um processo abrangente de geração de pesos, considerando toda a informação de entrada Chauhan et al. (2024).

2.9 Convolutional Neural Networks (CNN)

Uma rede neural é um tipo de algoritmo de ML projetado para simular o comportamento do cérebro humano. É composto por nós interconectados, também conhecidos como neurônios artificiais, que são organizados em camadas. As redes neurais são algoritmos de ML, mas diferem do aprendizado de máquina tradicional em vários aspectos importantes.

Elas aprendem e melhoram por si mesmas, sem intervenção humana. Elas aprendem recursos diretamente dos dados, tornando-o mais adequado para grandes conjuntos de dados. No entanto, no aprendizado de máquina tradicional, os recursos são fornecidos manualmente.

As redes neurais tradicionais, como o MLP, podem não ser apropriadas para problemas com dados que apresentam muitas variáveis, além de apresentarem ineficientes no tratamento de dados que possuam algum tipo de estrutura inerente, como relações espaciais. Para ambas situações pode ser necessário um número elevado de pesos a serem ajustados, já que um neurônio se conecta às entradas de todos os neurônios da camada seguinte. Isso pode provocar um aumento significativo no tempo de treinamento necessário, além de exigir um número muito grande de amostras para ajustar os parâmetros corretamente. Quando a segunda situação não é satisfeita, a rede além de não ser capaz de aprender algumas variações relevantes desses dados, como também fica sujeita a sobreajuste. Além disso, a estrutura de alguns dados como imagens, por exemplo, pode resultar em regiões da rede ajustadas para valores iguais LeCun, Bengio et al. (1995), o que além de um treinamento desnecessário pode causar ineficiência no armazenamento dos parâmetros da rede.

As Redes Neurais Convolucionais (*Convolutional Neural Networks* – ConvNets – CNNs) definem estruturas que contornam tais problemas e, consequentemente, a sua aplicação já pode ser considerada uma tendência em redes de computadores. A adoção em redes é simplificada pela existência de algumas técnicas que convertem os dados provenientes das redes de computadores em formatos apropriados para o uso em CNNs Sharma et al. (2019). Duas características das redes neurais convolucionais se destacam: menor número de parâmetros e resistência a variações nos dados, como translação e distorção. A esparsidade de conexões entre os neurônios de camadas adjacentes e o compartilhamento de parâmetros são fundamentais para tornar mais eficiente a retirada de características locais dos dados. Dessa forma, torna-se mais difícil que apenas pedaços da rede neural se especializem em uma tarefa enquanto outros pedaços são subutilizados. A esparsidade é obtida limitando o número de conexões que um neurônio pode fazer, isto é, cada neurônio recebe apenas saídas de uma pequena vizinhança de neurônios da camada anterior Goodfellow, Bengio e Courville (2016).

As CNN consiste em duas partes:

- um codificador convolucional que consiste em duas camadas convolucionais; e
- um bloco denso que consiste em três camadas totalmente conectadas;

As unidades básicas em cada bloco convolucional são uma camada convolucional, uma função de ativação sigmóide e uma subsequente operação média de *pooling*. Observe que, embora *ReLUs* e *max-pooling* funcionem melhor, essas descobertas ainda não haviam sido feitas na década de 1990. Cada camada convolucional usa um *kernel* 5×5 e uma função de ativação sigmóide. Essas camadas mapeiam entradas organizadas espacialmente a uma série de mapas de recursos bidimensionais, normalmente aumentando o número de canais. A primeira camada convolucional tem 6 canais de saída, enquanto o segundo tem 16. Cada operação de *pooling* 2×2 (passo 2) reduz a dimensionalidade por um fator de 4 por meio da redução da resolução espacial. O bloco convolucional emite uma saída com forma dada por (tamanho do lote, número de canal, altura, largura).

Para passar a saída do bloco convolucional para o bloco denso, devemos nivelar cada exemplo no *minibatch*. Em outras palavras, pegamos essa entrada quadridimensional e a transformamos na entrada bidimensional esperada por camadas totalmente conectadas: como um lembrete, a representação bidimensional que desejamos usa a primeira dimensão para indexar exemplos no *minibatch* e o segundo para dar a representação vetorial plana de cada exemplo. O bloco denso do *LeNet* tem três camadas totalmente conectadas, com 120, 84 e 10 saídas, respectivamente. Porque ainda estamos realizando a classificação, a camada de saída de 10 dimensões corresponde ao número de classes de saída possíveis.

Chegar ao ponto em que você realmente entende o que está acontecendo dentro do *LeNet* pode dar um pouco de trabalho, mas espero que o seguinte *snippet* de código o convença que a implementação de tais modelos com estruturas modernas de *deep learning* é extremamente simples. Precisamos apenas instanciar um bloco sequencial e encadear as camadas apropriadas.

2.10 Redes Neurais Pré-treinadas

As redes neurais pré treinadas podem acelerar o desenvolvimento e economizar tempo na execução de tarefas, elas podem facilitar utilização das técnicas de FL e TL, pois seu modelo pré treinado diminui o esforço que deveria ser realizado para os testes e treino de um modelo que começa a ser treinado do zero. Os exemplos de redes neurais pré treinadas são :

- ResNet - A *Residual Network* é uma arquitetura revolucionária da CNN que introduziu conexões de pulo para mitigar o problema do gradiente que desaparece.

- VGG16 - Desenvolvido pelo Grupo de Geometria Visual da Universidade de *Oxford*, o *VGG-16* é um modelo CNN amplamente adotado conhecido por sua simplicidade e eficácia.

- Início-v3 - Enfatiza a eficiência computacional e a extração eficaz de recursos. Introduzido pelo Google, este modelo utiliza módulos iniciais com múltiplas camadas convolucionais paralelas de diferentes tamanhos.

- MobileNet - A Rede móvel concentra-se em alcançar um equilíbrio entre o tamanho do modelo e a precisão. V. EfficientNet - Representa um avanço recente na arquitetura da CNN que aborda o desafio do dimensionamento de modelos.

2.10.1 VGG-16

A rede neural convolucional *VGG-16* tem sido amplamente usada na classificação de imagens, sua arquitetura específica e suas características permitem sua utilização conciliada com as técnicas de aprendizagem por transferência. A *VGG-16* foi proposta por um grupo de pesquisadores (geometria visual grupo) da Universidade de Oxford Sharafaldin, Lashkari e Ghorbani (2018).

A *VGG-16* é uma das arquiteturas do VGG, a outra é a *VGG-19*. A *VGG-16* possui 16 camadas, enquanto a *VGG-19* possui 19 camadas. A *VGG-16* contém principalmente três partes diferentes: padrão convolução, *pooling* e camadas totalmente conectadas. A arquitetura começa com duas camadas de convolução seguidas de *pooling*, depois outras duas convoluções seguidas de *pooling*, após aquela repetição de três convoluções seguidas por *pooling* e, finalmente, três camadas totalmente conectadas.

2.10.2 ResNet-50

O ResNet-50 é uma versão da rede residual de 50 camadas que consiste em 5 estágios, onde cada estágio inclui um bloco convolucional e um bloco de identidade. Cada bloco convolucional contém três camadas convolucionais padrão, e cada bloco de identidade contém três camadas convolucionais. Após os cinco estágios, uma camada de *pooling* médio para subamostragem, uma camada achatada, uma camada totalmente conectada e uma camada *softmax* foram adicionadas para classificação.

Vale lembrar que a ResNet-50 é uma versão da rede pré-treinada ResNet que contém uma arquitetura profunda (152 camadas) para classificação de imagens e foi vencedora do desafio do ImageNet de 2015.

2.11 Federated averaging (FedAvg)

O método *FedAvg* é o mais utilizado para implementar o Aprendizado Federado, pois seu objetivo é treinar modelos capazes de realizar previsões precisas com base em um conjunto de entradas, mesmo quando os dados de treinamento estão distribuídos entre vários dispositivos de borda.

Uma das principais vantagens do *FedAvg* é a possibilidade de realizar treinamento colaborativo sem a necessidade de transferir os dados armazenados nos dispositivos de borda

para um servidor central. Essa característica é essencial para garantir a privacidade dos dados e reduzir a sobrecarga da rede, tornando a abordagem mais eficiente e segura.

Além disso, ao executar dois modelos da mesma inicialização aleatória e treiná-los separadamente em subconjuntos com dados diferentes, McMahan McMahan et al. (2017), percebeu que *FedAvg* alcança um desempenho surpreendente, demonstrando sua eficácia na aprendizagem distribuída.

2.12 Frameworks de Aprendizagem Federada

Para facilitar a implementação da Aprendizagem Federada existem *Frameworks* que realizam a orquestração necessária para utilização da tecnologia. Os *Frameworks* são capazes de gerenciar o processo de transferência do modelo global para os dispositivos de bordas, bem como o fluxo inverso, do dispositivo de borda para um servidor central, além disso, o *Framework* deve ser responsáveis pela atividade de agregação de modelos que deve ocorrer no servidor central. Embora o uso de Aprendizagem Federada seja bastante recente, existem diversas bibliotecas *Python* para o desenvolvimento de suas aplicações. Podemos destacar alguns *Frameworks* de Aprendizagem Federada que se destacam no cenário atual:

2.12.1 Nvidia Flare

NVFlare é uma estrutura de Aprendizagem Federada (NVIDIA, 2024) pronta para negócios da *Nvidia*. Ele serve como espinha dorsal para *Nvidia Clara*, um pacote de produtos de IA para aplicativos de saúde, e tornou-se de código aberto em 2021. Sua quantidade de recursos bem projetados (relevantes para segurança e governança) e uma arquitetura reforçada para segurança são pontos positivos a serem considerados Shi et al. (2022).

2.12.2 TFF federado do TensorFlow

O *TensorFlow Federated* ou TFF McMahan et al. (2017) foi desenvolvido para expandir os modelos *TensorFlow/Keras* existentes para a configuração federada. A Google criou uma estrutura de código aberto para métodos de ML aplicados a dados descentralizados, o TFF suporta o *Tensorboard* e permite algoritmos de agregação personalizados, criado para facilitar pesquisas e experimentações abertas usando FL. Foi selecionado para o presente trabalho por conter vantagens específicas, compatibilidade com o problema abordado e características que atendem aos requisitos do Aprendizado Federado como: APIs dedicadas para treinar modelos em dados descentralizados; suporte a estratégias personalizadas de agregação e otimização; permite simular ambientes federados reais com facilidade, que incluem a distribuição de dados entre clientes, treinamento local seguido de agregação global e a avaliação de estratégias em cenários controlados.

2.12.3 IBM FL Community Edition

A IBM também criou sua própria estrutura FL IBM Research (2024) , que chamou de *IBM Federated Learning*. Esta é uma biblioteca projetada para oferecer suporte a uma implementação fácil de ML em um ambiente federado. Essa estrutura é uma edição da comunidade orientada à pesquisa do serviço FL da IBM. Ele suporta vários modelos, incluindo *clustering* federado e árvores de decisão, e até implementa algoritmos de trabalhos de pesquisa recentes

2.12.4 PySyft

É uma biblioteca OpenMined (2024) de código aberto baseada em *Python* 3 que permite Aprendizado Federado para propósitos de pesquisa e utiliza FL, desenvolvida pela comunidade *OpenMined* funciona principalmente com *frameworks* de aprendizado profundo como *PyTorch* e *TensorFlow*. O *PySyft* se concentra em fornecer ao FL comunicação segura e privada.

2.12.5 Flower - "Friendly Federated Learning Framework"

É uma biblioteca Flower Authors (2024) estável de alto nível para *Python*, originou-se de um projeto de pesquisa na Universidade de *Oxford*, e agora é gerenciado pela adap gmbh, uma empresa alemã de IA distribuída. Devido ao seu histórico, *Flower* torna relativamente simples personalizar, estender ou substituir componentes. *Flower* ajuda na transição rápida de implementações de ML existentes para uma configuração de FL. Isto permite uma forma rápida de avaliação dos modelos existentes num ambiente federado e simples de implementar, suportando tanto experimentos básicos quanto configurações avançadas permitindo flexibilidade e compatibilidade com *TensorFlow*, *PyTorch*, *Keras*, *Scikit-learn* Beutel et al. (2020), sendo principal motivo para a escolha de integrar o presente trabalho.

Percebe-se que foram estudadas diferentes ferramentas para o referente trabalho, escolhendo as que melhor se adequavam para com a proposta.

2.13 DevOps

Conjunto de práticas que combina desenvolvimento de software (Dev) e operações de TI (Ops). O *DevOps* visa encurtar o ciclo de vida de desenvolvimento do sistema, ao mesmo tempo em que fornece recursos, correções e atualizações frequentemente em estreito alinhamento com os objetivos de negócios. No centro da filosofia *DevOps* estão as práticas de Integração Contínua (CI) e Entrega Contínua (CD), que formam a espinha dorsal de uma implementação *DevOps* bem sucedida. Integração Contínua é uma prática de desenvolvimento onde os desenvolvedores integram código em um repositório compartilhado frequentemente, de preferência várias vezes ao dia. Cada integração é verificada por uma compilação automatizada

e testes automatizados para detectar erros de integração o mais rápido possível. Essa prática ajuda a garantir que o *software* esteja sempre em um estado que pode ser lançado, reduzindo significativamente o tempo necessário para entregar novos recursos e correções Banala (2024).

O movimento *DevOps* transformou o cenário de desenvolvimento e entrega de *software*. Em sua essência, o *DevOps* busca integrar equipes de desenvolvimento e operações para melhorar a colaboração, agilizar processos e acelerar a entrega de software de alta qualidade Banala (2024).

3 Metodologia

A metodologia descreve os itens que compõem o projeto dissertativo, após um aprofundamento do referencial teórico e uma análise exaustiva, foram definidos os conjuntos de dados utilizados, bem como os classificadores (modelos) que mais se adequam ao projeto de pesquisa.

Os conjuntos de dados foram treinados utilizando um classificador previamente parametrizado e ajustado. Posteriormente, esse mesmo classificador foi empregado no treinamento de outros conjuntos de dados, a princípio com as mesmas parametrizações. Além disso, foram selecionados conjuntos de dados compatíveis com a técnica de FL, levando em consideração características essenciais como privacidade dos dados do usuário, descentralização e colaboração.

A seguir, elencamos as etapas, técnicas e estratégias de preparação utilizadas para os testes e treinamento realizados durante o desenvolvimento do projeto.

3.1 Conjunto de dados

Visando obter os resultados esperados para os testes e treinamento que possa ser utilizado com as técnicas propostas de TL e FL, foram selecionados dois conjuntos de dados com as seguintes características :

A escolha de conjunto de dados que possuam características recomendadas pela academia, foi preponderante para o trabalho:

- Inclui registros de auditoria e dados brutos de rede;
- Contém variedade de ataques modernos;
- Representa tráfego normal realista e diversificado;
- Está rotulado;
- Fornece garantia de privacidade;
- É aceito pela comunidade.

3.1.1 BOT-IoT

O conjunto de dados BOT-IoT foi criado para auxiliar na detecção de intrusão em redes IoT. Os pacotes de rede brutos (arquivos Pcap) do conjunto de dados BOT-IoT foram criados pela ferramenta *tshark*, no *Cyber Range Lab do Australian Center for Cyber Security* (ACCS), ele incorpora um combinação de tráfego normal e anormal.

O BOT-IoT foi criado projetando um ambiente de rede realista no *Cyber Range Lab da UNSW Canberra*. O ambiente de rede incorporou uma combinação de tráfego normal e de botnet (rede de computadores infectados que podem ser controlados remotamente). Os arquivos de origem do conjunto de dados são fornecidos em diferentes formatos, incluindo os arquivos pcap originais, os arquivos argus gerados e os arquivos csv. Os arquivos foram separados, com base na categoria e subcategoria do ataque, para melhor auxiliar no processo de rotulagem.

O tráfego de rede simulado foi gerado através da ferramenta Ostinato (ferramenta para gerar trafego e medição de rede) e *Node-red* (para não-IoT e IoT, respectivamente). O conjunto de dados inclui ataques *DDoS*, *DoS*, OS e *Service Scan*, *Keylogging* e ataques de exfiltração de dados (transferência intencional não autorizada de dados de um dispositivo para outro), com os ataques *DDoS* e *DoS* ainda mais organizados, com base no protocolo usado. As amostras geradas para representar o tráfego malicioso corresponde Classe Positiva, enquanto a Classe Negativa é representada pelas amostras que corresponde ao tráfego normal.

3.1.2 TON_IoT

O conjunto de dados TON_IoT, também projetado pelo laboratório Australiano UNSW Canberra, pertence a nova geração de conjuntos de dados da Indústria 4.0, IoT e IIoT para avaliar a fidelidade e a eficiência de diferentes aplicações de segurança cibernética baseadas em Inteligência Artificial (IA), ou seja, algoritmos de ML e DL.

O conjuto de dados recebeu esse nome porque incluem fontes de dados heterogêneas coletadas de conjuntos de dados de telemetria de sensores IoT e IIoT, conjuntos de dados de sistemas operacionais do *Windows* 7 e 10, bem como conjuntos de dados *Ubuntu* 14 e 18 TLS e tráfego de rede. Os conjuntos de dados foram coletados de uma rede realista e em grande escala projetada no *Cyber Range e IoT Labs*, na Escola de Engenharia e Tecnologia da Informação (SEIT), *UNSW Canberra na Australian Defense Force Academy* (ADFA).

Uma nova rede de teste foi criada para a rede da indústria 4.0 que inclui redes IoT e IIoT. A plataforma de teste foi implantada usando múltiplas máquinas virtuais e hosts de sistemas operacionais *Windows*, *Linux* e *Kali* para gerenciar a interconexão entre as três camadas de sistemas IoT, *Cloud* e *Edge/Fog*. Várias técnicas de ataque, como *DoS*, *DDoS* e *ransomware*, contra aplicações web, *gateways* IoT e sistemas informatizados em toda a rede IoT/IIoT. Os conjuntos de dados foram reunidos em um processamento paralelo para coletar vários eventos normais e de ataque cibernético do tráfego de rede, rastreamentos de auditoria do *Windows*, rastreamentos de auditoria do *Linux* e dados de telemetria de serviços IoT Moustafa (2019). Da mesma forma que ocorre no conjunto de dados BOT-IoT a classe positiva é representada pelo tráfego malicioso e a classe negativa é representada pelo tráfego normal de rede.

3.2 Pré-processamento e separação dos dados

Os dados extraídos em sua forma original nem sempre estão prontos para serem utilizados por um classificador, sendo necessária a aplicação de técnicas de pré-processamento para normalização e ajuste. O pré-processamento é uma etapa essencial da mineração de dados, responsável por transformar informações brutas em um formato adequado para análise Fung, Sandilya e Rao (2005).

No mundo real, os dados frequentemente apresentam inconsistências, lacunas e ruídos, além de possíveis erros que comprometem a qualidade do treinamento. Dessa forma, o pré-processamento é fundamental para mitigar esses problemas.

No presente trabalho, foram realizadas etapas de limpeza, incluindo preenchimento de valores ausentes, suavização de dados ruidosos e eliminação de redundâncias, ainda durante essa etapa foi realizada a transformação dos rótulos originais em uma classificação binária. O tráfego normal foi rotulado como classe negativa (0), enquanto qualquer tipo de ataque ou comportamento anômalo foi rotulado como classe positiva (1). Esta abordagem facilita a aplicação de técnicas supervisionadas de ML, permitindo ao modelo aprender a distinguir entre padrões normais e maliciosos.

Após a limpeza, os dados foram integrados e normalizados, garantindo homogeneidade entre as diferentes representações. Em seguida, foram divididos em conjunto de treino e conjunto de teste, prática amplamente utilizada na literatura, também conhecida como conjunto de validação. Essa etapa permite avaliar o desempenho do modelo em dados não vistos, simulando sua performance real. Para este estudo, adotou-se a estratégia *train-test-split*, utilizando 80% dos dados para modelagem e 20% para teste Tan et al. (2021).

Para a execução do FL, o conjunto de dados foi particionado de acordo com o número de clientes, determinando o tamanho de cada amostra. A função train-test-split da biblioteca *Scikit-learn* foi utilizada para auxiliar na separação dos dados, garantindo uma distribuição equilibrada entre os participantes.

A metodologia adotada neste trabalho associa as técnicas de FL e TL, utilizando o algoritmo *FedAvg* como função de agregação e LNN como modelo base. Essa escolha é respaldada por estudos anteriores, como os de Gazeau, Gupta e An (2024), Vinayakumar et al. (2019).

O FL foi implementado em duas abordagens distintas:

- Abordagem centralizada: utilizando um servidor FL responsável pela coordenação do processo de aprendizado e agregação dos modelos locais.
- Abordagem descentralizada: envolvendo 10 clientes FL, cada um processando seus dados localmente, sem a dependência de um servidor central para coordenação direta.

A implementação foi realizadas com o *framework Flower*, garantindo flexibilidade e escalabilidade do modelo. Além disso, técnicas de DevOps foram incorporadas para orquestrar a TL, otimizando a implementação e garantindo segurança e automação no processo de atualização dos modelos.

3.3 Modelos

No contexto da execução de tarefas cotidianas, é possível traçar um paralelo com as técnicas abordadas nesta dissertação —FL e TL. Ambas podem ser comparadas ao processo de transferência de conhecimento adquirido em uma atividade ou tarefa já conhecida para a realização de uma nova tarefa, de natureza semelhante. Essa reutilização de experiências anteriores facilita a adaptação e a execução eficiente de novas demandas. Além disso, é possível considerar que tais atividades podem ser realizadas por pessoas diferentes, em escopos distintos, e em locais variados, exigindo, ainda assim, o cumprimento de requisitos como confiabilidade, integridade, entre outros requisitos.

Ao reutilizar o conhecimento adquirido em atividades similares, torna-se mais fácil alcançar êxito na execução de novas tarefas que compartilhem características semelhantes. A busca por um modelo eficaz que utilize as técnicas de TL e FL será realizada ao longo da execução dos experimentos desenvolvidos neste projeto dissertativo.

Para a escolha do algoritmo, foram consideradas as tendências identificadas nos artigos que compõem a Revisão Sistemática da Literatura deste trabalho. Além disso, foi feito o acompanhamento de trabalhos recentes publicados em mídias diversas, como transmissões ao vivo (streams) e conteúdos na internet.

Na aplicação da técnica de Aprendizado Federado, associada ao Aprendizado por Transferência, utilizou-se o *FedAvg* como função de agregação, e modelos LNN foram escolhidos com base em tendências observadas em publicações recentes Gazeau, Gupta e An (2024), Vinayakumar et al. (2019).

A técnica abordada para métrica foi FL com LNN no formato centralizado (servidor FL) e para o formato descentralizado (10 clientes FL) utilizamos LNN e Flower com *FedAvg*, buscando por um modelo eficaz para utilizar a técnica FL avaliando a execução durante os experimentos desenvolvidos no trabalho. Além disso, foram incorporadas práticas de *DevOps* para viabilizar o processo de transferência de conhecimento (TL) de forma estruturada e automatizada.

3.4 Desenvolvimento do modelo

A arquitetura foi projetada para diagnosticar de forma binária a distinção entre as amostras que apresentam padrões e comportamentos benignos ou maliciosos, utilizando um

modelo de regressão logística, foram descritas as etapas de processamento que devem ser realizadas até a etapa de treinamento, a arquitetura proposta está representada na Figura 12. Iniciando pela etapa de recepção dos dados, eles são obtidos através dos dois conjuntos de dados com características com algumas similaridades.

Após a recepção do conjunto de dados externo, é necessário realizar uma etapa de normalização para garantir a qualidade dos dados utilizados no treinamento dos modelos. Nesse processo, amostras inconsistentes ou problemáticas devem ser identificadas e removidas, evitando ruídos que possam comprometer o desempenho do modelo. Essa etapa é fundamental para minimizar riscos de sobreajuste (*Overfitting*), garantindo que o modelo aprenda padrões generalizáveis e mantenha um alto nível de precisão em novos dados.

Na próxima etapa aplica-se a técnica de FL, é realizado o treinamento do modelo LNN de forma centralizada do lado do servidor com *PyTorch* e o *Framework Flower* derivando o modelo binário. Utilizando como base o modelo binário inicial é gerado o modelo global, que um dos itens principal da técnica FL. Nesta etapa ainda deve ocorrer a geração modelo TL para ser utilizado através da estrutura de *DevOps*, conforme definição da arquitetura FTL.

Na última etapa de processamento o modelo global é transferido para as organizações, onde serão treinadas de forma descentralizada, utilizando o *Flower framework* com *FedAvg*.

3.4.1 Descrição e transformação de dados

Para desenvolver o IDS proposto foram utilizados dois conjuntos de dados públicos que possuem amostras com diversos tipos de ataques, vale apenas ressaltar que os conjuntos de dados selecionados possuem amostras com ataques atualizados. Os conjuntos TON_IoT e o BOT_IoT se destacam por possuírem amostras específicas de ataques em ambientes IoT.

O conjunto de dados BOT-IoT, possuem amostras que podem ser coletadas em uma rede não-IoT e IoT, além das amostras benignas, os principais ataques que podem ser identificados no conjunto de dados são : *DDoS*, *DoS*, OS e *Service Scan* e *Keylogging*.

O processo de transformação de dados foram detalhados utilizando os conjuntos de dados BOT-IoT e TON_IoT, os padrões de ataque encontrados noa conjuntos de dados

Os conjuntos de dados originais geralmente possuem algumas amostras que precisam ser removidas ou ajustadas para que não ocorra desvio durante as etapas de testes e treinos, para o nosso estudo, o tratamento das amostras precisam ser realizadas para que não ocorra erros durante a conversão dos dados tabulados em imagem. Como parte da fase de pré-processamento de dados, os conjuntos de dados são limpos de todos os valores de dados ausentes ou mal informados. Os conjuntos de dados foram verificados em busca de amostras que não continham valores e valores inadequados como nan, inf, -inf, etc. Essas amostras foram descartadas devido ao grande volume do conjunto de dados.

Utilizando recursos disponíveis na biblioteca *Sklearn* foram realizadas os seguintes procedimento para deixar os conjunto de dados TON_IoT e o BOT_IoT pronto para processamento :

- Eliminar os espaço antes de cada nomes das features
- Atribuir 0 para cada características não negativos quando houver valores negativos
- Remover colunas com variação igual a zero (tem apenas 1 valor exclusivo)
- Remover inf, -inf, nan e linhas duplicadas
- Eliminar colunas com valores idênticos

Após o processo de limpeza dos dados, é necessário realizar uma etapa de pré-processamento sobre o conjunto de dados resultante, a fim de gerar entradas adequadas para o modelo proposto.

Considerando que os modelos LNN funcionam melhor em conjuntos de dados tabulares — uma vez que são compostos essencialmente por camadas densas (totalmente conectadas), sem convoluções ou mecanismos especializados para captar estruturas espaciais —, sua utilização se mostra apropriada para o tratamento de dados de tráfego de redes, que são, em sua maioria, estruturados em formato tabular Hussain et al. (2020).

3.4.2 Descrição do Gráfico:

O gráfico figura 12 ilustra o fluxo do processo de métrica, dividido em etapas distintas:

3.4.2.1 Servidor Central (Processo Centralizado)

A primeira etapa apresenta o servidor central, armazenado na nuvem, responsável pela execução do processo centralizado de treinamento inicial dos modelos.

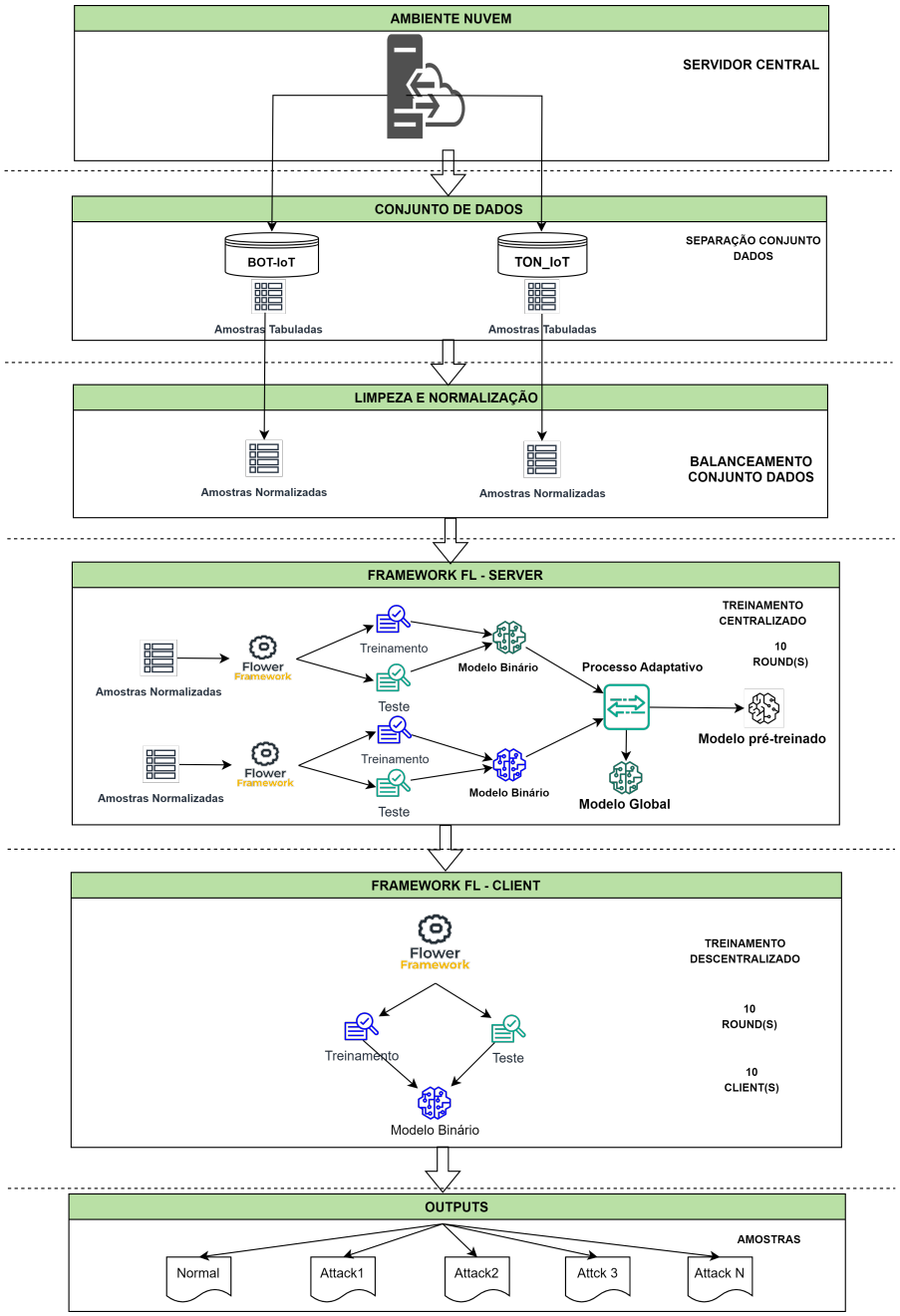
3.4.2.2 Seleção e Pré-processamento dos Dados

Em seguida, ocorre a seleção dos conjuntos de dados públicos, seguida pela etapa de normalização, que inclui processos de limpeza e balanceamento dos dados para garantir sua qualidade e representatividade.

3.4.2.3 Treinamento Centralizado e Geração dos Modelos

A terceira etapa aborda o treinamento centralizado, onde são gerados dois modelos principais: o Modelo Global e o Modelo de Transferência (pré-treinado), que serão utilizados em cenários descentralizados.

Figura 12 – Proposta para Otimização Modelo LNN.



Fonte: Adaptdao de Yang e Shami (2022).

O gráfico ilustra o ciclo completo do processo de desenvolvimento dos modelos, desde o treinamento centralizado até a execução descentralizada utilizando o *Framework Flower*, culminando na geração dos resultados finais.

3.4.2.4 Treinamento Descentralizado

A etapa seguinte foca no treinamento descentralizado, realizado nos clientes por meio do *Framework Flower*, que possibilita a aplicação do Aprendizado Federado.

3.4.2.5 Outputs e Resultados

Por fim, o gráfico apresenta os *Outputs* gerados pelo processo, destacando as amostras classificadas e os resultados obtidos a partir dos modelos treinados.

3.4.3 Modelo Linear proposto

Conforme descrito na figura 13, o modelo proposto contém três camadas lineares totalmente conectadas e duas funções de ativação, adequado para utilização em conjuntos de dados tabulado. Iniciando pela *Input* é definido pelo número de características no seu conjunto de dados que será utilizado para realizar o treinamento, é realizado o mapeamento para 128 neurônios. A segunda camada linear do modelo é definida de forma oculta, nela é realizada a redução de 128 para 64 neurônios . A última camada do modelo, que também podemos chamar de camada de saída, a dimensão de 64 neurônios é reduzida para 1, produzindo o resultado final.

As funções de ativação escolhidas foram *ReLU* (*Rectified Linear Unit*), atualmente a mais utilizada em projetos redes neurais, é usada após as duas primeiras camadas lineares, uma das suas funções é transformar valores negativos em zero, promovendo uma ativação esparsa na rede.

A definição matemática da função *ReLU* é a seguinte:

$$f(x) = max(0, x) \tag{3.1}$$

- Valor Positivo: Se a entrada x for maior que 0, a função retorna x.
- Valor Negativo: Se a entrada x for menor ou igual a 0, a função retorna 0

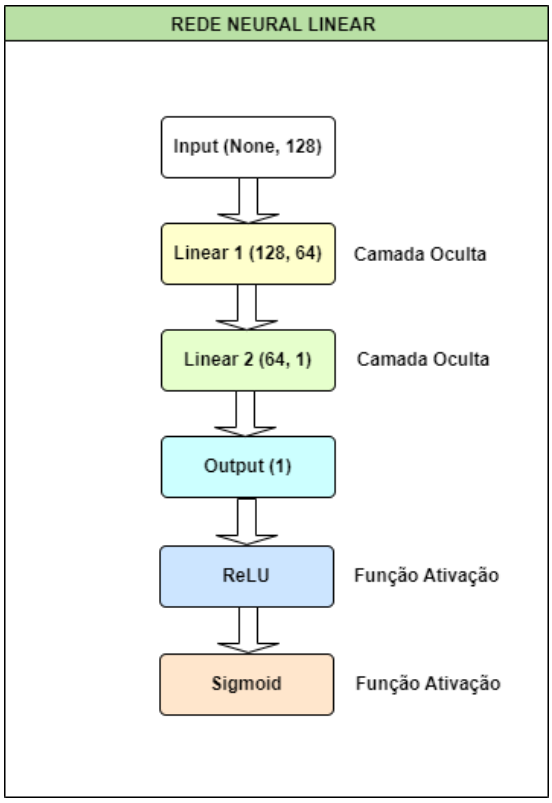
A outra função de ativação utilizada no modelo é a sigmóide utilizada após a camada de saída, restringindo a saída entre 0 e 1, apropriada para uma tarefa de classificação binária. A maior vantagem sobre a função de etapa e a função linear é que não é linear. Esta é uma característica incrivelmente interessante da função sigmóide.

3.4.4 LNN - Redes Neurais Lineares e aprendizagem por transferência

A escolha do modelo LNN, também conhecido como Rede Neural Densa ou *Fully Connected*, deve-se às características relevantes observadas em comparação com outros modelos de ML. A LNN é um modelo de DL amplamente utilizado em tarefas de classificação sobre conjuntos de dados tabulares, devido à sua estrutura simples e eficiente.

Embora modelos como as CNNs sejam atualmente mais utilizados e apresentem desempenho superior em conjuntos de imagens, no contexto deste trabalho — que envolve a

Figura 13 – Modelo rede neural linear proposta



Fonte: Elaborado pelo autor.

A figura exibe de forma detalhada as camadas lineares e ocultas que constituem o modelo *Linear Neural Network* (LNN) utilizado na pesquisa, destacando a arquitetura composta por camadas de entrada, camadas ocultas e camada de saída, além das funções de ativação aplicadas ao longo do processo de treinamento.

classificação de tráfego de rede para IDS — os dados utilizados são, predominantemente, tabulares. Dessa forma, não se justifica a aplicação de redes convolucionais, uma vez que não há estruturas espaciais a serem exploradas. Além disso, observou-se que modelos lineares apresentam desempenho competitivo, ou até superior, quando aplicados a esse tipo de dado.

Para modelos DL, a técnica de TL consiste em um o processo de transferência dos pesos de um modelo DNN treinado em um conjunto de dados para outro conjunto de dados Leonardo et al. (2018). A técnica TL foi aplicada com sucesso em tarefas de processamento de imagens. Isso ocorre porque os padrões de recursos aprendidos pelas camadas inferiores dos modelos CNN são geralmente padrões gerais aplicáveis a muitas tarefas diferentes, e apenas os recursos aprendidos pelas camadas superiores são recursos específicos para um conjunto de dados específico Leonardo et al. (2018). Portanto, as camadas inferiores dos modelos CNN podem ser transferidas diretamente para diferentes tarefas. Para melhorar a eficácia do TL, o ajuste fino pode ser usado no processo de TL dos modelos DL. No ajuste fino, a maioria das camadas do modelo pré-treinado são congeladas (ou seja, seus pesos são retidos), enquanto algumas das camadas superiores são descongeladas para treinar novamente o modelo em um novo conjunto de dados. O ajuste fino permite que o modelo de aprendizagem atualize

os recursos de ordem superior no modelo pré treinado para melhor se adequar à tarefa ou conjunto de dados Leonardo et al. (2018).

Essa mesma técnica será aplicada em modelos LNN, após o treinamento de um modelo em um determinado conjunto de dados, o mesmo modelos será utilizado para treinar outro conjunto de dados.

3.4.5 Critérios de avaliação de desempenho do modelo

Para realizar os primeiros experimentos, foi utilizado um microcomputador PC para etapa inicial dos estudos, a configuração básica não permitia uma ótima performance, o código foi executado em no computador com a seguinte conficuração: CPU Intel® Core™ i7-1255U (10-core, cache de 12MB, até 4.7GHz), *Windows* 11 Home, Português 64 bits e GPU Intel® Iris® Xe com memória gráfica compartilhada e memória RAM 16GB DDR4 (2x8GB). Com relação aos recursos de desenvolvimento, utilizou-se linguagem de programação *Python* e diversos componentes como: *Numpy*, *Pandas* e *Matplotlib*, outro recurso utilizado para facilitar facilitar o desenvolvimento, foi a utilização da biblioteca *scikit-learn*, que é uma bibliotecas de aprendizado de máquina de código aberto. A biblioteca *PyTorch* foi utilizada para ganho de produtividade na implementação, criação e treinamento de redes neurais de aprendizado profundo DL.

A proposta do modelo é avaliada através dos dois conjuntos de dados , BOT-IoT e TON_IoT Yang e Shami (2022), conforme descrito na seção 3.1. A validação cruzada quíntupla é utilizada para avaliar o modelo proposto, que pode evitar o sobreajuste e o resultados tendenciosos.

Os conjuntos de dados gerados a partir do tráfego de rede frequentemente apresentam desbalanceamento significativo, caracterizado por uma predominância de amostras rotuladas como "Normais"e uma proporção relativamente pequena de amostras que representam ataques. Esse desequilíbrio pode comprometer a capacidade dos modelos de aprendizado de máquina de detectar ameaças com precisão, tornando essencial a utilização de métricas complementares para avaliar seu desempenho de forma adequada.

Além da avaliação da eficácia na detecção de ataques, a eficiência computacional do modelo também foi analisada. Foram mensurados o tempo de treinamento e o tempo de teste, ambos executados em um ambiente computacional padrão (PC). Essa análise permite verificar a viabilidade da implementação do modelo em cenários reais, garantindo que o tempo de resposta seja adequado para aplicações de segurança cibernética em tempo real.

3.5 Arquitetura de Aprendizado Federado por Transferência

A arquitetura proposta tem como objetivo permitir a detecção precisa de ataques maliciosos à rede por meio do aprendizado federado por transferência, preservando a privacidade e a segurança dos dados institucionais. Para isso, emprega-se um modelo baseado em LNNs pré-treinadas, que é compartilhado e refinado entre diferentes núcleos organizacionais e dispositivos de computação de borda, sem comprometer sua capacidade de generalização.

Na estrutura apresentada, os dispositivos periféricos — como computadores de borda e dispositivos IoT — interagem com K organizações (representando diferentes usuários) e um servidor em nuvem, o qual é responsável por consolidar o modelo global. Essa abordagem permite a expansão para um número maior de organizações em cenários reais, garantindo flexibilidade e escalabilidade ao sistema. A Figura 14 ilustra a visão geral da arquitetura proposta.

A Figura 14 ilustra um exemplo de comunicação entre as organizações — Organização 1, Organização 2 e OrganizaçãoK — que, devido à existência de um "muro" de proteção relacionado à privacidade de dados, não podem compartilhar diretamente suas próprias informações. Seus respectivos conjuntos de dados são representados por D1, D2 e DK, e os modelos correspondentes são Modelo 1, Modelo 2 e Modelo$K$, respectivamente.

Cada organização está impedida de utilizar os dados ou modelos de outras organizações para fins de treinamento. A única forma de troca de conhecimento ocorre por meio de um modelo global, hospedado na nuvem, por meio do processo de migração de conhecimento.

Assim, cada modelo local se comunica exclusivamente com o modelo global, o que possibilita a quebra da "parede de separação" entre as organizações e viabiliza o treinamento colaborativo, sem que sejam violadas as restrições de privacidade dos dados.

A estrutura protege a privacidade dos dados de várias instituições e pode usar o conhecimento do modelo de cada organização para ajudar a treinar seus próprios modelos. A estrutura inclui principalmente os quatro procedimentos a seguir:

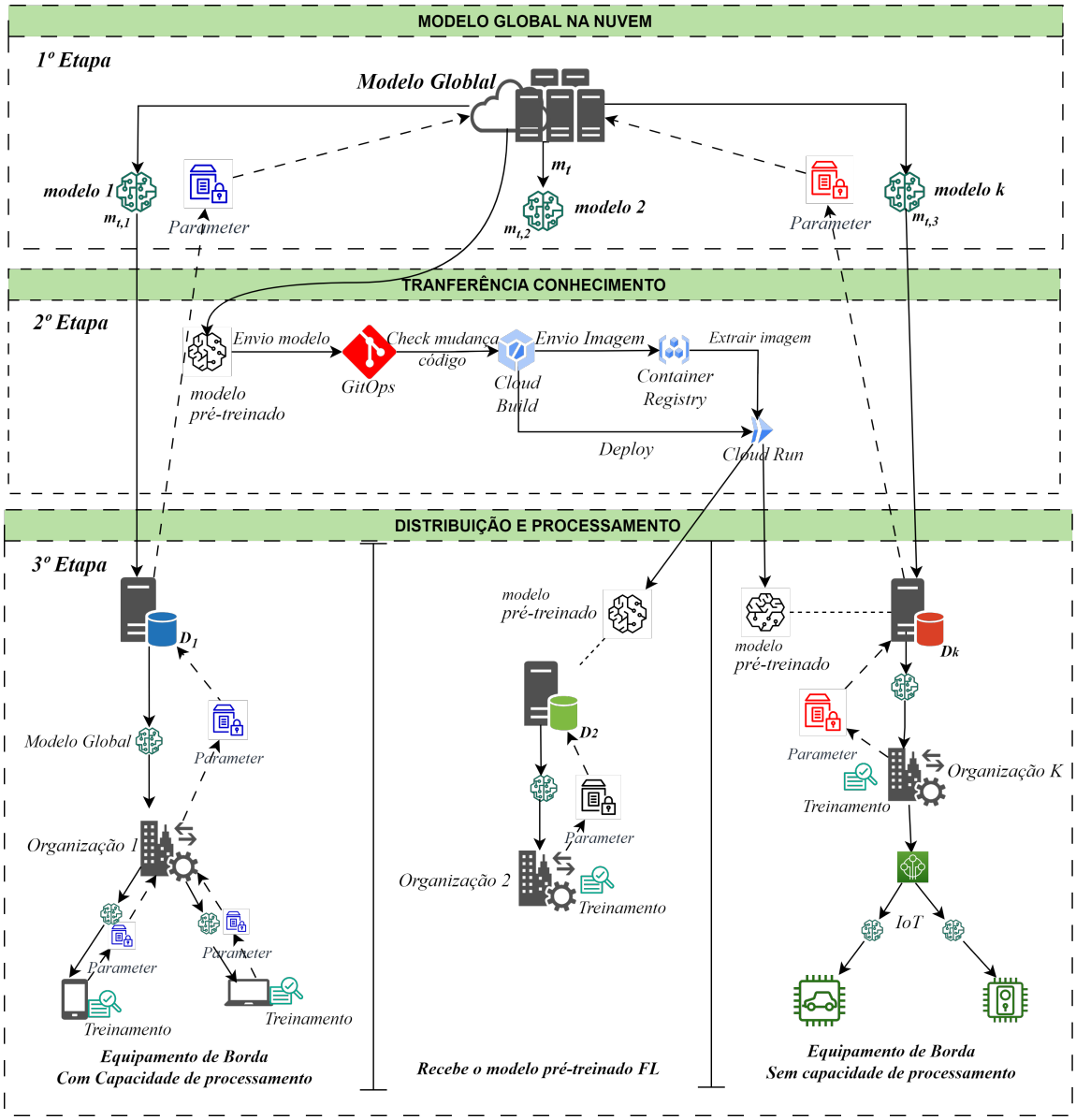
1) O modelo global disponível no servidor na nuvem é incialmente treinado de acordo com o conjunto de dados externo;

2) O servidor distribui o modelo global para todas as instituições, cada instituição pode treinar o modelo antes de distribuir na sua rede, gerando seu próprio modelo com seus próprios dados (a Figura 13 mostra o processo de migração de conhecimento);

3) O modelo de cada organização é transferido para o servidor na nuvem, deve passar por uma nova série de treinamento e com base nos valores obtidos, deve ser submetido a um novo *emsemble*;

4) O procedimento deve ser repetido de uma a três vezes, até que a convergência seja alcançada e cada organização possa obter um modelo personalizado com um bom efeito de aprendizado.

Figura 14 – Proposta para Arquitetura FTL



Fonte: Adaptdao de Yang e Shami (2022).

A figura ilustra a arquitetura FTL, estruturada em três etapas principais: a primeira apresenta o modelo global armazenado na nuvem e o processo de troca de parâmetros; a segunda descreve a *Pipeline DevOps*, responsável pelo *deploy* automatizado do modelo pré-treinado; e a terceira destaca dois cenários de processamento descentralizado utilizando FL, evidenciando as interações entre os clientes e o servidor central.

O processo de aprendizado dos modelos envolve o compartilhamento de parâmetros. Após a criação do modelo global armazenado na nuvem é estabelecido, ele pode ser reutilizado por diferentes organizações.

Vale ressaltar que os experimentos desta dissertação foram realizados utilizando a funcionalidade de simulação de múltiplos clientes do *Flower Framework*, através desta funcioanlidade um único computador pode ser utilizado para devidir o conjunto de dados proporcionalmente e simular a quantidade de clentes necessário para reproduzir um cenário fiel do

FL.

O *Flower Framework* foi escolhido por sua flexibilidade e capacidade de gerenciar ambientes descentralizados de forma eficiente. Além disso, o *Flower* se destaca por sua fácil integração com diversos *frameworks* de aprendizado de máquina, como *TensorFlow*, *PyTorch* e *Keras*, permitindo a implementação de soluções robustas e escaláveis. Sua arquitetura modular possibilita a customização dos algoritmos de treinamento federado, adaptando-se tanto a cenários de pesquisa quanto a aplicações em produção.

3.5.1 Critérios de avaliação para a arquitetura Federated Transfer Learning

Para verificar a eficácia e a capacidade de generalização do algoritmo proposto na detecção de intrusões da arquitetura FTL, utilizaram-se dois conjuntos de dados de detecção de intrusão — BOT-IoT e TON_IoT — empregados tanto para treinamento quanto para testes do modelo global, sendo adotados como conjuntos experimentais.

As principais métricas de avaliação utilizadas com frequência em sistemas de detecção incluem ACC, *Precision*, *Recall* e *F1-Score* .

A *Precision* representa a proporção de amostras corretamente classificadas como positivas em relação ao total de amostras classificadas como positivas. Nesse cálculo, as amostras classificadas como falsos positivos (FP) também são consideradas. Quanto maior o número de verdadeiros positivos (TP), maior será o valor da precisão.

- TP (Verdadeiro Positivo): Representa o número de amostras positivas que são classificadas corretamente como amostras positivas;
- TN (Verdadeiro Negativo):Representa o número de amostras negativas classificadas corretamente como amostras negativas
- FP (Falso Positivo): Representa o número de amostras negativas classificadas incorretamente como amostras positivas.
- FN (Falso Negativo): Representa o número de amostras positivas classificadas corretamente como amostras negativas.

Dessa forma, a taxa média de *Precision* e a taxa de falsos alarmes obtidas por meio dos resultados experimentais da técnica de Validação Cruzada (*Cross Validation*), com 10 execuções, são utilizadas como indicadores de avaliação geral para medir a eficácia e a precisão do algoritmo.

A técnica de Validação Cruzada consiste em dividir o conjunto de dados de treinamento em n subconjuntos (ou partições), permitindo uma avaliação mais robusta do desempenho do modelo ao longo de múltiplos testes.

4 Resultados

Os resultados obtidos indicam um desempenho satisfatório em todos os cenários analisados, considerando tanto a aplicação de técnicas de FL centralizado e descentralizado quanto o treinamento de modelos pré-treinados para TL.

Para a validação e avaliação do modelo global, foram utilizados os conjuntos de dados BOT-IoT e TON_IoT. Além disso, o modelo pré-treinado empregado na técnica de TL foi submetido a testes rigorosos para garantir sua eficácia. A avaliação do desempenho preditivo dos modelos foi conduzida de maneira imparcial, assegurando a confiabilidade dos resultados.

É importante destacar que os experimentos realizados em cenários descentralizados utilizaram o método de simulação do *Flower*, o que permitiu uma análise detalhada das métricas relevantes e evidenciou o desempenho do modelo em ambientes federados.

A análise detalhada dos experimentos foi realizada por meio da construção de gráficos e tabelas que apresentam as métricas empregadas durante o treinamento e os testes, conforme descrito a seguir:

- ACC (*Accuracy*) - Precisão global do modelo;
- AUC (Área Sob a Curva ROC) - Capacidade do modelo em distinguir entre classes;
- *Loss* (Função de Perda) - Medida do erro do modelo durante o treinamento;
- *Precision* - Precisão das previsões para a classe positiva;
- *Recall* - Sensibilidade/Cobertura, representando a taxa de acertos para a classe positiva.

Nos testes realizados em um ambiente centralizado, os valores de ACC obtidos ao longo das épocas demonstraram resultados superiores a 0.99%. Já nos testes conduzidos em um ambiente descentralizado, a análise do gráfico da função de perda (*Loss*) revelou uma maior variação no conjunto de dados, com oscilações de até 0.55%. Além disso, foi mensurado o tempo de processamento de cada modelo, assim como os valores da métrica AUC.

Outro aspecto relevante identificado nos experimentos foi o comportamento do *Recall*, que apresentou valores extremamente baixos, próximos de zero. Esse resultado evidencia uma alta taxa de falsos negativos, indicando que o modelo teve dificuldades em identificar corretamente amostras da classe positiva.

Por fim, a Tabela 6 apresenta uma comparação entre os diferentes tipos de ataques presentes nos conjuntos de dados BOT-IoT e TON_IoT, destacando suas similaridades e diferenças.

Para a separação dos conjuntos de dados, empregou-se a função *train_test_split()* da

Tabela 8 – Distribuição amostral por conjuntos de dados utilizados nos experimentos

Categoria	TON_IoT	TON_IoT (50%)	TON_IoT (5%)	TON_IoT (1%)	BOT_IoT	BOT_IoT (10%)
Benign	6.099.469	1.800.561	180.024	36.200	135.037	13.525
Scanning	3.781.419	1.500.498	150.062	29.843	-	-
XSS	2.455.020	1.224.643	122.668	24.501	-	-
DDoS	2.026.234	874.654	87.434	17.406	18.331.847	1.833.465
Password	1.153.323	496.852	49.471	10.077	-	-
DoS	712.609	326.600	32.658	6.491	16.673.183	1.667.249
Injection	684.465	330.471	33.096	6.545	-	-
Backdoor	16.809	8.063	806	155	-	-
MITM	7.723	3.912	399	109	-	-
Ransomware	3.425	1.686	176	31	-	-
Reconnaissance	-	-	-	-	2.620.999	261.886
Theft	-	-	-	-	2.431	224
Total	**16.940.496**	**6.567.940**	**656.794**	**131.358**	**37.763.497**	**3.776.348**

biblioteca *scikit-learn*, garantindo uma divisão proporcional das amostras contidas nos conjuntos. Essa abordagem visa minimizar potenciais vieses no processo de avaliação e validação dos modelos, assegurando que a distribuição das amostras seja representativa e equilibrada.

A geração do modelo global e do modelo pré-treinado foi realizada utilizando o critério de 50% das amostras do conjunto de dados TON_IoT para o modelo global, esse modelo é utilizado como base para os demais treinamento e geração ou melhoria dos modelos. Para a geração do modelo pré-treinado utilizamos a proporção de 10% das amostras do conjunto BOT-IoT. Essa divisão foi definida com base em uma característica fundamental do TL, na qual o modelo deve ser treinado utilizando um conjunto de dados que contenha menor quantidade de amostras. Embora o conjunto BOT-IoT possua um número significativamente maior de amostras, a proporção destinada à geração do modelo pré-treinado foi menor, mantendo a consistência da abordagem de TL e garantindo a generalização do modelo.

A Tabela 6 apresenta a quantidade de amostras selecionadas após o processo de divisão e normalização dos dados, evidenciando a distribuição final utilizada no treinamento dos modelos.

4.1 Resultados dos Testes Centralizados - Modelo Global

O treinamento do modelo global foi iniciado utilizando o conjunto de dados TON_IoT. Após o processo de normalização, 50% do conjunto de dados passou a conter 6.567.940 amostras, conforme detalhado na Tabela 6. Destaca-se que esse conjunto de dados possui 42 características, sendo essencial que o modelo linear seja treinado respeitando essa dimensão.

Desde a primeira rodada de treinamento, os indicadores alcançados nas métricas ACC, AUC, *Precision* e *Recall* demonstraram-se altamente estáveis. Os valores obtidos foram 0.9999 para ACC e AUC, e 1.0 para *Precision* e *Recall*. Esse desempenho pode ser atribuído ao processo de normalização das amostras realizado na etapa anterior, garantindo uma distribuição mais uniforme dos dados de entrada.

Os resultados obtidos na métrica ACC reforçam a assertividade do modelo, uma vez que foi aplicado em um conjunto de dados balanceado. O alto percentual de acertos sugere que o modelo está atingindo o desempenho esperado. No entanto, apesar de a métrica AUC ter apresentado um valor ótimo, ela não pode ser considerada um fator determinante para avaliação do modelo neste caso específico, pois a distribuição balanceada dos dados minimiza a necessidade dessa métrica como critério principal.

A métrica *Loss* apresentou uma variação maior no conjunto de dados de teste. Essa oscilação já era esperada, visto que conjuntos menores de dados tendem a apresentar um valor de *Loss* mais elevado, pois o modelo dispõe de menos amostras para aprendizado. Além disso, essa variação pode indicar a ocorrência de *overfitting* ou a falta de representatividade de algumas classes no conjunto de teste. O menor e o maior valor de *Loss* registrados no conjunto de teste foram, respectivamente, 0.0015 e 0.0858, representando a maior variação observada.

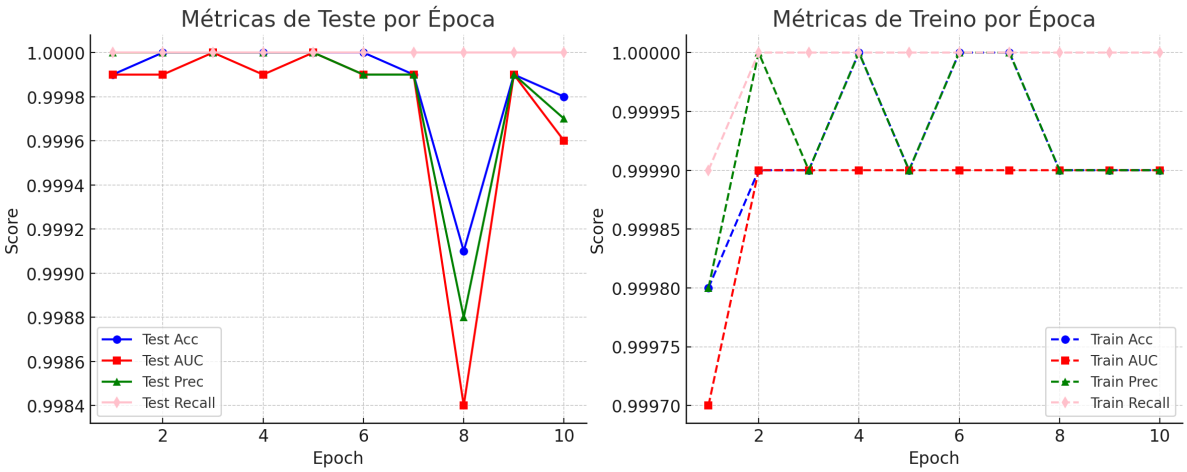
As métricas *Precision* e *Recall* também apresentaram indicadores elevados desde a primeira época, atingindo valores de 1.0 para ambas. Um alto valor de *Precision* durante o treinamento sugere que a quantidade de Falsos Positivos (FP) foi mínima, o que pode ser verificado por meio da matriz de confusão.

Ao comparar as métricas ACC, AUC, *Loss*, *Precision* e *Recall* no treinamento do modelo aplicado ao conjunto de dados de teste, observa-se que todos os indicadores apresentaram resultados excelentes. A alta ACC e AUC indicam que o modelo está separando corretamente as classes e realizando previsões precisas. Além disso, a baixa *Loss* reforça que o modelo está aprendendo adequadamente, ajustando suas previsões de maneira eficaz. Os valores de *Precision* e *Recall* igual a 1.0 evidenciam a capacidade do modelo em classificar corretamente os Verdadeiros Positivos (TP) e identificar um número reduzido de Falsos Positivos (FP), garantindo alta assertividade nas previsões.

Tabela 9 – Resultados do Modelo por Época

Epoch	Test Loss	Test Acc	Test AUC	Test Prec	Test Recall	Confusion Matrix
1	0.0032	0.9999	0.9999	1.0000	1.0	[[360235, 43], [27, 953283]]
2	0.0030	1.0000	0.9999	1.0000	1.0	[[360255, 23], [42, 953268]]
3	0.0015	1.0000	1.0000	1.0000	1.0	[[360263, 15], [20, 953290]]
4	0.0036	1.0000	0.9999	1.0000	1.0	[[360237, 41], [12, 953298]]
5	0.0022	1.0000	1.0000	1.0000	1.0	[[360260, 18], [17, 953293]]
6	0.0043	1.0000	0.9999	0.9999	1.0	[[360221, 57], [3, 953307]]
7	0.0045	0.9999	0.9999	0.9999	1.0	[[360205, 73], [7, 953303]]
8	0.0858	0.9991	0.9984	0.9988	1.0	[[359150, 1128], [5, 953305]]
9	0.0047	0.9999	0.9999	0.9999	1.0	[[360222, 56], [10, 953300]]
10	0.0186	0.9998	0.9996	0.9997	1.0	[[359976, 302], [2, 953308]]

Figura 15 – Gráficos - Treinamento e Teste Centalizado - TON_IoT



Fonte: Elaborado pelo autor.

O primeiro gráfico apresenta os resultados das métricas ACC, AUC, *Loss*, *Precision* e *Recall* obtidos na execução do modelo sobre o conjunto de dados de teste. O segundo gráfico exibe os mesmos indicadores, referentes à execução do modelo no conjunto de dados de treinamento, com 50% das amostras normalizadas do TON_IoT e de forma centralizadas, originando o modelo global.

4.2 Resultados Testes centralizados - Modelo Pré-Treinado

Para a geração do modelo pré-treinado, foi reutilizado o modelo global previamente treinado com o conjunto de dados TON_IoT. Esse processo ocorre a partir da utilização de 10% das amostras normalizadas do conjunto de dados BOT-IoT, que apresenta diferenças tanto nas amostras quanto em algumas de suas características.

Inicialmente, o modelo global foi treinado utilizando um conjunto de dados composto por 42 características. No entanto, para a etapa de aprendizado por transferência TL, o conjunto BOT-IoT foi adotado, contendo 44 características. Essa diferença na dimensionalidade dos dados exige uma adaptação na estrutura do modelo, especificamente na camada de entrada da LNN.

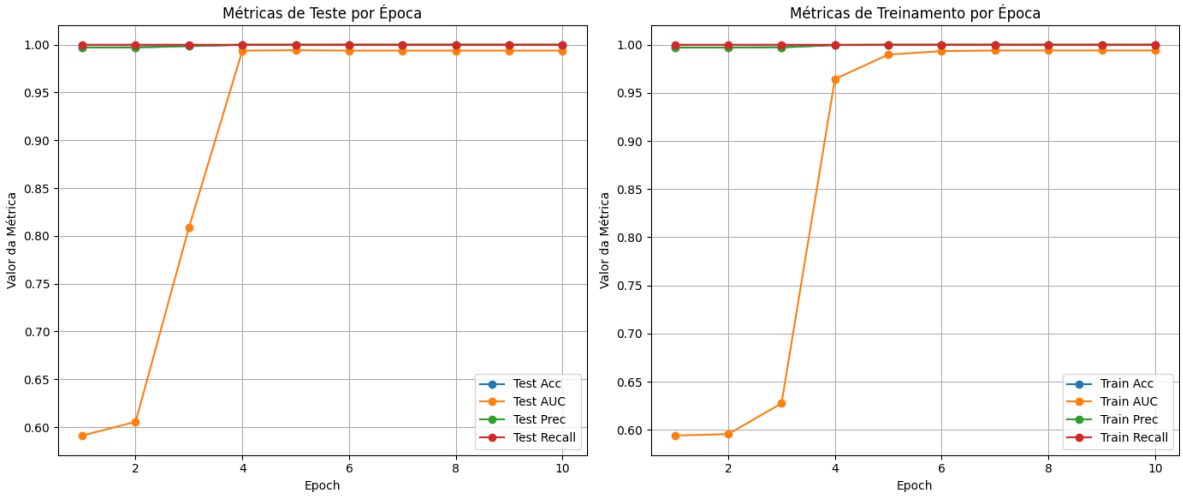
Para permitir essa compatibilidade, o modelo passa por uma camada intermediária responsável pelo redimensionamento das características, ajustando a entrada de 44 para 42 dimensões. Dessa forma, o modelo consegue ser treinado em um novo conjunto de dados sem comprometer as características previamente aprendidas. Esse processo é um elemento fundamental da técnica de TL, possibilitando o reaproveitamento do conhecimento adquirido no treinamento anterior e garantindo maior generalização do modelo.

Os resultados obtidos demonstram que mesmo reutilizado em espaços de caracteristicas diferentes, as métricas : ACC, *Precision* e *Recall* se mantém com excelentes resultads, o que não ocorre para a métrica *Loss*. A métrica AUC apresenta baixos resultados até a terceira epóca, para as demais apresentam bons resultados.

Tabela 10 – Resultados do Modelo por Época

Epoch	Test Loss	Test Acc	Test AUC	Test Prec	Test Recall	Confusion Matrix
1	0.2988	0.9970	0.5912	0.9971	0.9999	[[490, 2196], [61, 752523]]
2	0.2888	0.9971	0.6055	0.9972	0.9999	[[567, 2119], [62, 752522]]
3	0.1460	0.9985	0.8088	0.9986	0.9999	[[1659, 1027], [81, 752503]]
4	0.0224	0.9998	0.9938	1.0000	0.9998	[[2653, 33], [136, 752448]]
5	0.0199	0.9998	0.9942	1.0000	0.9998	[[2655, 31], [119, 752465]]
6	0.0187	0.9998	0.9938	1.0000	0.9999	[[2653, 33], [109, 752475]]
7	0.0187	0.9998	0.9938	1.0000	0.9999	[[2653, 33], [109, 752475]]
8	0.0187	0.9998	0.9938	1.0000	0.9999	[[2653, 33], [109, 752475]]
9	0.0187	0.9998	0.9938	1.0000	0.9999	[[2653, 33], [109, 752475]]
10	0.0187	0.9998	0.9938	1.0000	0.9999	[[2653, 33], [109, 752475]]

Figura 16 – Gráficos - Treinamento e Teste Centalizado - BOT-IoT



Fonte: Elaborado pelo autor.

O primeiro gráfico apresenta os resultados das métricas ACC, AUC, *Loss*, *Precision* e *Recall* obtidos na execução do modelo pré-treinado sobre o conjunto de dados de teste. O segundo gráfico exibe os mesmos indicadores referentes à execução do modelo pré-treinado sobre o conjunto de dados de treinamento. Ambos os experimentos foram realizados utilizando o conjunto de dados normalizados BOT-IoT, com 10% das amostras.

Para a execução dos testes, foi utilizado o *Framework Flower*, que possibilita a simulação da execução simultânea do processamento distribuído entre os 10 clientes.

Observou-se, ainda, uma melhora significativa nos valores das métricas a partir da quarta época, a estabilização do *Loss* sugere que o modelo está aprendendo de forma eficiente e não ao apresenta sinais de oscilação, indicando estabilidade nos resultados. Especificamente, a métrica ACC apresentou um desempenho inicial alto, atingindo 0.9970 já na primeira execução. Ao longo das épocas apresentou pouca oscilação e se mantendo com o indice de 0.9998 a partir da quarta época.

A métrica AUC apresentou a maior variação, situando-se entre 0.5912 a menor e 0.9942

a melhor na execução do teste para a quinta época, permanecendo estável em 0.9938 para os demais. No caso da *Precision*, a primeira época já atingiu um ótimo resultado 0.9971, e apresenta pouca oscilação. A partir da quarta época a métrica manteve-se constante em 1.0. O *Recall* apresentou comportamento semelhante, mantendo-se estável em 0.9999 para quase todas as épocas, com pequenas oscilações entre 0.9998 e 0.9999.

A matriz de confusão evidencia um índice extremamente baixo de Falsos Positivos (FP) e Falsos Negativos (FN), reforçando a precisão do modelo.

Nas primeiras épocas, o modelo apresenta um número elevado de Falsos Positivos (FP) e poucos (Verdadeiro Negativo (TN), indicando que muitos exemplos negativos são classificados como positivos. Com o avanço do treinamento, especialmente a partir da terceira época, há um aumento significativo dos TN e uma redução drástica dos FP. A partir da época 4, os valores se estabilizam. O modelo passa a classificar quase todos os negativos corretamente, com FP próximos de zero e TN muito elevados. Essa estabilidade indica que o treinamento convergiu e os resultados se mantêm consistentes nas épocas subsequentes.

A redução dos FP e a manutenção de altos valores de verdadeiro Positivo (TP) elevam os índices de ACC, AUC e *Precision*. Mesmo com um leve aumento nos Falsos Negativos (FN) em algumas épocas, o *Recall* permanece próximo de 1, demonstrando que o modelo consegue identificar a grande maioria dos exemplos positivos. A seguir, apresenta-se tabela 8 com a comparação desses valores:

Tabela 11 – Resultados do Modelo por Época - Descentralizado Modelo Global - 1º Cliente

Epoch	Test Loss	Test Acc	Test AUC	Test Prec	Test Recall	Confusion Matrix
1	0.0114	0.9999	0.9999	1.0000	0.9998	[[7175, 0], [3, 19094]]
2	0.0152	0.9998	0.9999	0.9999	0.9998	[[7174, 1], [3, 19094]]
3	0.0152	0.9998	0.9999	0.9999	0.9998	[[7174, 1], [3, 19094]]
4	0.0121	0.9998	0.9999	1.0000	0.9998	[[7175, 0], [4, 19093]]
5	0.0152	0.9998	0.9999	0.9999	0.9998	[[7174, 1], [3, 19094]]
6	0.0121	0.9998	0.9999	1.0000	0.9998	[[7175, 0], [4, 19093]]
7	0.0152	0.9998	0.9999	0.9999	0.9998	[[7174, 1], [3, 19094]]
8	0.0121	0.9998	0.9999	1.0000	0.9998	[[7175, 0], [4, 19093]]
9	0.0152	0.9998	0.9999	0.9999	0.9998	[[7174, 1], [3, 19094]]
10	0.0027	0.9999	0.9999	1.0000	0.9999	[[7175, 0], [2, 19095]]

4.3 Resultados dos Testes Descentralizados - Modelo Global

A partir do modelo global gerado na Seção 4.1, foram conduzidos testes e treinamentos de forma descentralizada, utilizando o conjunto de dados TON_IoT ao longo de 10 épocas para 10 clientes.

Inicialmente, os testes seriam realizados sobre o conjunto de dados normalizado na proporção de 5%, totalizando 656.794 amostras. No entanto, verificou-se que os recursos de hardware disponíveis eram insuficientes para a execução desse volume de dados. Diante dessa limitação, optou-se por reduzir a proporção para 1%, resultando em um total de 131.358 amostras após a normalização.

Para a execução dos testes, foi utilizado o *Framework Flower*, que possibilita a simulação da execução simultânea do processamento distribuído entre os 10 clientes.

Os resultados obtidos demonstram que, mesmo com a reutilização do modelo global, as métricas ACC, *Precision*, *Recall*, *Loss* e AUC mantiveram-se com excelentes desempenhos.

Observou-se, ainda, uma constância nos valores das métricas a partir do segundo cliente, indicando estabilidade nos resultados. Especificamente, a métrica ACC apresentou um desempenho inicial elevado, atingindo 0.9999 já na primeira execução. Ao longo das épocas, oscilou entre 0.9998 e 1.0 para o primeiro cliente, enquanto para os demais manteve-se constante em 0.9999.

A métrica AUC apresentou ainda menor variação, situando-se entre 0.9999 e 1.0 na execução dos testes para o primeiro cliente e permanecendo estável em 0.9999 para os demais.

No caso do *Precision*, o primeiro cliente já atingiu o valor máximo de 1.0, oscilando levemente para 0.9999 em algumas execuções. Para os demais clientes, a métrica manteve-se constante em 1.0 ao longo de todas as épocas.

O *Recall* apresentou comportamento semelhante, mantendo-se estável em 0.9999 para todos os clientes e em todas as épocas, com pequenas oscilações entre 0.9998 e 0.9999 apenas no primeiro cliente.

A matriz de confusão também evidencia um índice extremamente baixo de Falsos Positivos (FP) e Falsos Negativos (FN), reforçando a precisão do modelo. A seguir, apresenta-se uma tabela comparativa desses valores:

Tabela 12 – Resultados do Modelo por Época - 2º à 10º Clientes

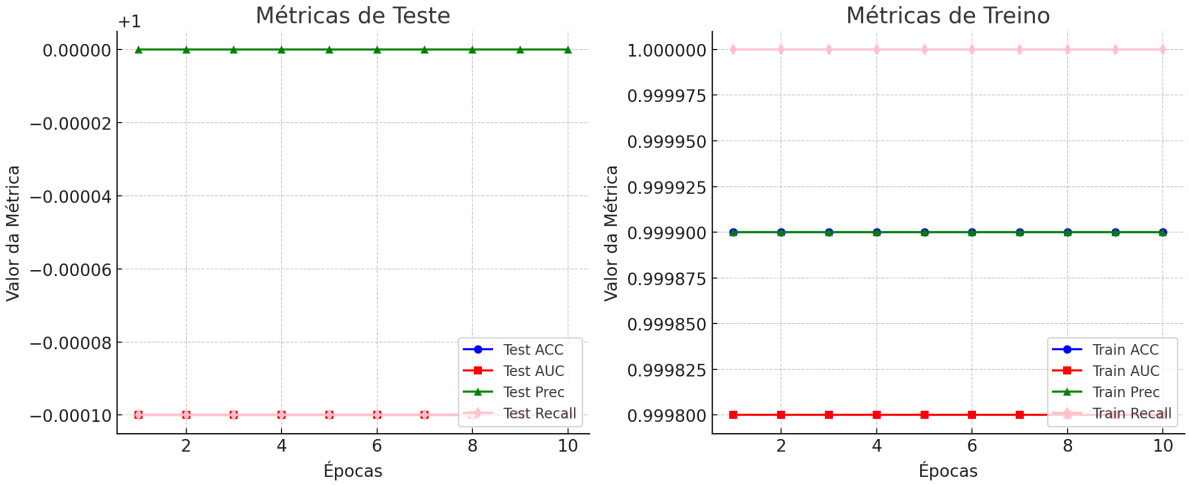
Epoch	Test Loss	Test Acc	Test AUC	Test Prec	Test Recall	Confusion Matrix
1	0.0027	0.9999	0.9999	1.0000	0.9999	[[7175, 0], [2, 19095]]
2	0.0027	0.9999	0.9999	1.0000	0.9999	[[7175, 0], [2, 19095]]
3	0.0027	0.9999	0.9999	1.0000	0.9999	[[7175, 0], [2, 19095]]
4	0.0027	0.9999	0.9999	1.0000	0.9999	[[7175, 0], [2, 19095]]
5	0.0027	0.9999	0.9999	1.0000	0.9999	[[7175, 0], [2, 19095]]
6	0.0027	0.9999	0.9999	1.0000	0.9999	[[7175, 0], [2, 19095]]
7	0.0027	0.9999	0.9999	1.0000	0.9999	[[7175, 0], [2, 19095]]
8	0.0027	0.9999	0.9999	1.0000	0.9999	[[7175, 0], [2, 19095]]
9	0.0027	0.9999	0.9999	1.0000	0.9999	[[7175, 0], [2, 19095]]
10	0.0027	0.9999	0.9999	1.0000	0.9999	[[7175, 0], [2, 19095]]

Figura 17 – Gráficos - Treinamento e Teste Descentralizado - TON_IoT

Fonte: Elaborado pelo autor.

Realizamos a demonstração do ciclo completo de execução do primeiro *round*, equivalente à realização dos testes e do treinamento efetuados para o primeiro cliente na estrutura descentralizada. O primeiro gráfico apresenta os resultados das métricas ACC, AUC, *Loss*, *Precision* e *Recall* obtidos na execução do modelo global reutilizado sobre o conjunto de dados de teste. O segundo gráfico exibe os mesmos indicadores referentes à execução do modelo global sobre o conjunto de dados de treinamento. Ambos os experimentos foram realizados utilizando o conjunto de dados normalizados TON-IoT, com 1% das amostras.

Figura 18 – Gráficos - Treinamento e Teste Descentralizado - TON_IoT



Os dois gráficos apresentam os resultados das métricas ACC, AUC, *Loss*, *Precision* e *Recall* obtidos na execução do modelo global reutilizado nos conjuntos de dados de teste e de treinamento. Os experimentos foram realizados utilizando o conjunto de dados TON-IoT, com 1% das amostras normalizadas e particionado em 9 rounds, correspondentes a 9 clientes. A visualização foi consolidada em virtude da similaridade dos resultados das métricas.

4.4 Comparando as métricas ACC, AUC, *LOSS*, *Precision* e *Recall* com relação aos testes

Há diversas combinações que pode ser realizada para comparar as métricas ACC, AUC, *LOSS*, *Precision* e *Recall*, selecionamos a comparação que se adequa melhor ao estudo realizado.

- ACC alta e AUC alta:

  Modelo está separando e prevendo corretamente;

- ACC alta e AUC baixa:

  O modelo pode estar ignorando a classe minoritária;

- ACC baixa e AUC alta:

  Bom modelo com limiar de decisão mal ajustado;

- ACC baixa e AUC alta :

  O modelo tem uma boa capacidade de separação entre classes, mas o limiar de decisão pode estar mal ajustado;

- LOSS baixo e AUC alto:

  Modelo está aprendendo bem e separando as classes.

- LOSS baixo e AUC baixo:

  Modelo está fazendo predições confiantes, mas não separa bem as classes.

- LOSS baixo e ACC alta:

  Modelo ajustado e previsões corretas na maior parte dos casos.

- LOSS baixo e ACC baixa:

  Modelo está aprendendo, mas pode haver desequilíbrio de classes ou erro na separação.

- *Precision* alta e *Recall* alta:

  O modelo está realizando previsões altamente confiáveis, com baixa incidência de FP e FN.

- *Precision* alta e *Recall* baixa:

  Esse cenário demostra uma fragilidade do modelo, pois aponta um número alto FN.

- *Precision* baixa e *Recall* alta:

  O modelo identifica a maioria das ameaças, baixo número de FN, porém alto número FP;

- *Precision* baixa e *Recall* baixa:

  O modelo possui baixo desempenho geral, com um número elevado tanto de falsos positivos quanto de falsos negativos.

A comparação entre as métricas *Precision* e *Recall* com foco em segurança, destacando suas implicações em sistemas críticos, como detecção de fraudes, segurança cibernética.

4.4.1 Análise comparativa das métricas entre modelos

A análise comparativa das métricas de desempenho dos modelos treinados permite avaliar a eficiência de diferentes abordagens de aprendizado de máquina. Neste estudo, foram comparados modelos centralizados e descentralizados, bem como modelos pré-treinados e treinados globalmente. As métricas consideradas incluem ACC, AUC, *Loss*, *Precision* e *Recall*, permitindo uma compreensão aprofundada do comportamento dos modelos em diferentes cenários.

4.4.2 Comparação entre modelos centralizados e descentralizados

O modelo centralizado global apresentou valores de ACC e AUC extremamente altos, próximos a 1.0000, demonstrando que o modelo consegue separar corretamente as classes. No entanto, ao analisar a métrica de *Loss*, observa-se uma pequena variação ao longo das épocas, o que pode indicar um ajuste contínuo do modelo ao conjunto de dados. Em contrapartida, o modelo centralizado pré-treinado exibiu um desempenho inferior, com AUC de aproximadamente 0.5912, indicando dificuldades na separação das classes, apesar da alta ACC. Esse comportamento sugere que o modelo pode estar ignorando a classe minoritária, tornando-se inadequado para cenários onde é fundamental identificar eventos raros, como em sistemas de segurança e detecção de fraudes.

No modelo descentralizado global, os resultados foram consistentes entre os diferentes clientes. O primeiro cliente apresentou *Loss* ligeiramente maior nas primeiras épocas, mas estabilizou-se, acompanhando o desempenho dos demais clientes. Isso indica que a abordagem descentralizada conseguiu manter a generalização do modelo ao longo das rodadas de treinamento. O alto valor de *Precision* e *Recall* reforça a confiabilidade das previsões, com baixa taxa de FP e FN, demonstrando um modelo eficiente para aplicações críticas, como segurança cibernética.

4.4.3 Impacto das métricas na qualidade do modelo

A relação entre *Loss* e AUC fornece uma visão sobre a capacidade de aprendizado do modelo. Nos modelos globais, o *Loss* manteve-se baixo enquanto o AUC permaneceu alto,

confirmando que o modelo foi capaz de aprender adequadamente e separar as classes de maneira eficiente. O modelo pré-treinado, por outro lado, apresentou *Loss* elevado e AUC baixo, indicando que suas previsões, embora confiantes, não conseguiram distinguir corretamente as classes.

A combinação de *Precision* e *Recall* também desempenha um papel crucial na avaliação da performance do modelo. Modelos com *Precision* e *Recall* altos são preferíveis para aplicações onde tanto a minimização de FP quanto de FN são essenciais, como em sistemas de segurança e monitoramento de fraudes. No caso do modelo global descentralizado, ambos os valores permaneceram elevados, reforçando sua robustez. Já modelo pré-treinado apresenta índices elevados para as métricas *Precision* e *Recall*. Observa-se que a métrica *Precision* sofre uma pequena oscilação, enquanto o *Recall* se mantém estável. Esse padrão sugere uma maior incidência de falsos negativos, o que pode comprometer a aplicabilidade do modelo em cenários críticos.

Com base na comparação das métricas, conclui-se que o modelo global centralizado e o modelo descentralizado apresentaram desempenho superior ao modelo pré-treinado, especialmente no que diz respeito à separação correta das classes e à confiabilidade das previsões. A alta ACC e AUC observadas nos modelos globais indicam que esses modelos são capazes de realizar previsões corretas e bem ajustadas à distribuição dos dados. Além disso, a estabilidade das métricas ao longo das épocas reforça a eficiência do treinamento contínuo.

Já o modelo pré-treinado, apesar de apresentar alta ACC, mostrou limitações na separação de classes, possivelmente ignorando instâncias da classe minoritária. Isso destaca a importância de ajustes finos nos limiares de decisão e na adaptação de modelos para diferentes domínios de aplicação. Em sistemas críticos, como segurança cibernética e detecção de fraudes, é fundamental priorizar modelos que combinem alta *Precision* e *Recall*, garantindo previsões confiáveis e minimizando erros de classificação.

4.4.4 Comparação entre Experimentos

Ao comparar os resultados obtidos no artigo *Federated Learning for IoT Intrusion Detection*, de autoria de Lazzarini, Tianfield e Charissis (2023), com os resultados alcançados nos experimentos realizados neste trabalho, observa-se que as métricas obtidas pelo modelo LNN proposto apresentam desempenho superior em relação aos valores apresentados no referido artigo. É importante destacar que ambos os estudos empregaram um modelo de classificação binária para a detecção de intrusões em ambientes IoT, o que torna a comparação direta e relevante.

Tabela 13 – Comparação do desempenho dos classificadores no conjunto de dados TON_IoT

2*Model	Lazzarini, Tianfield e Charissis (2023)			Modelo LNN Proposto		
	Acc	**Precision**	**Recall**	**Acc**	**Precision**	**Recall**
Client 1	0.9758	0.9449	0.9883	0.9999	1.0000	0.9998
Client 2	0.9755	0.9436	0.9892	0.9999	1.0000	0.9990
Client 3	0.9748	0.9440	0.9864	0.9999	1.0000	0.9990
Client 4	0.9760	0.9458	0.9878	0.9999	1.0000	0.9990
FLModel—Aggr.	0.9759	0.9487	0.9842			
Centralized	0.9840	0.9729	0.9817	0.9999	1.0000	1.0000

Comparação dos resultados entre o modelo de Lazzarini, Tianfield e Charissis (2023) e o modelo LNN proposto. No caso do modelo LNN, os valores apresentados correspondem ao desempenho obtido na primeira época de execução para cada cliente.

4.5 Transferência modelo pré-treinado

A transferência de um modelo pré-treinado deve ocorrer sob demanda do cliente. Nesse contexto, é fundamental que empresas e clientes estejam cientes de que estão adquirindo um modelo previamente treinado em outros conjuntos de dados. No momento da implantação, é essencial que o cliente tenha acesso às informações pertinentes, como a data de geração, o número da release e o certificado de assinatura, garantindo assim a rastreabilidade e a conformidade do aceite.

Após a conclusão do treinamento, conforme descrito no item 4.2, o modelo encontra-se apto para atualização. Nesse estágio, a geração do modelo é finalizada e torna-se necessário seu envio para o *GitOps*. Esse processo viabiliza a criação da release do modelo, permitindo um controle de versão eficiente, além de facilitar a colaboração, garantir conformidade e integrar práticas de CI/CD (*Continuous Integration/Continuous Deployment*).

Uma vez publicado na esteira de *deploy*, o modelo passa por uma verificação rigorosa antes de ser disponibilizado no *Cloud Build*. Essa etapa é crucial, pois o *Cloud Build* é responsável por importar o código-fonte, promover o modelo e realizar os testes necessários antes da implantação. Esse processo ocorre em repositórios ou espaços do *Cloud Storage*, culminando na geração e envio da imagem para o *Container Registry*. Vale destacar que os recursos do *Cloud Build* atendem aos requisitos do nível 3 dos Níveis da Cadeia de Suprimentos para Artefatos de Software (*Supply Chain Levels for Software Artifacts* - SLSA).

O *Container Registry* desempenha um papel central ao armazenar a imagem do modelo em um arquivo binário, garantindo uma orquestração eficiente e centralizada dos dados essenciais. Além disso, essa ferramenta permite que as equipes armazenem e gerenciem imagens de modelos de maneira escalável e segura, consolidando todas as versões em um único repositório virtualmente ilimitado.

Na etapa final, o modelo deve ser disponibilizado no *Cloud Run*, serviço que possibilita a execução gerenciada de aplicativos em contêineres na nuvem. No entanto, no presente caso, a execução ocorre de forma esporádica, visto que o objetivo principal é a transferência do modelo para outra organização ou cliente, assegurando a portabilidade e a flexibilidade do sistema.

A segurança da estrutura de *deploy* é um fator essencial para garantir a integridade, a confidencialidade e a disponibilidade do modelo pré-treinado ao longo de todo o seu ciclo de vida. Para isso, são implementadas boas práticas de proteção, incluindo autenticação robusta, controle de acesso baseado em funções (*Role-Based Access Control* - RBAC) e auditorias contínuas das operações realizadas.

No *Cloud Build*, cada etapa do processo de implantação é monitorada e registrada, assegurando rastreabilidade e conformidade com padrões de segurança. Além disso, a utilização do *Container Registry* garante a verificação de assinaturas digitais e a análise de vulnerabilidades em imagens de contêineres antes da implantação, prevenindo riscos de execução de código malicioso. A comunicação entre os serviços ocorre por meio de conexões seguras e criptografadas, reduzindo a exposição a ataques cibernéticos. Por fim, a disponibilização no *Cloud Run* segue protocolos rígidos de isolamento e execução segura, permitindo que o modelo seja transferido para outras organizações ou clientes sem comprometer a integridade dos dados e garantindo um ambiente seguro e controlado.

5 Conclusão e trabalhos futuros

A partir dos estudos e experimentos conduzidos, foi possível selecionar os modelos, conjuntos de dados e técnicas que nortearam o desenvolvimento deste trabalho. Para a base de dados, optou-se pelos conjuntos BOT-IoT e TON_IoT, que contêm amostras de intrusão geradas em dispositivos IoT. Em relação aos modelos, foi escolhido o modelo base de Redes Neurais LNN, implementado tanto em uma abordagem centralizada quanto descentralizada. O objetivo principal foi o desenvolvimento de uma arquitetura capaz de explorar os recursos da técnica FTL, ampliando a eficiência da detecção de intrusão em redes IoT.

A análise detalhada dos resultados demonstrou desempenho excepcional dos modelos de FL, independentemente da configuração utilizada. As métricas de ACC AUC próximas de 1.0, aliadas a baixos valores de *Loss*, evidenciam a capacidade dos modelos de aprender e generalizar os padrões dos dados de maneira eficiente. Além disso, o tempo de treinamento manteve-se em níveis aceitáveis, reforçando a viabilidade da abordagem para aplicações práticas em segurança cibernética.

Outro achado relevante foi o desempenho do *framework Flower*, que se destacou por sua flexibilidade e agilidade ao operar em ambientes centralizados e descentralizados. Essa característica reforça sua importância para projetos que demandam escalabilidade, segurança e confiabilidade, consolidando-o como uma ferramenta estratégica no avanço das Soluções de IDS baseadas em FL.

A arquitetura desenvolvida também buscou sanar uma lacuna presente nos *frameworks* atuais de FL, que, até o momento, não oferecem suporte nativo à orquestração da transferência de modelos pré-treinados para clientes ou organizações. O estudo demonstrou que uma *pipeline* de *DevOps* pode ser uma alternativa viável e inovadora para gerenciar esse processo de forma automatizada e segura. A aplicação das práticas de CI/CD, fundamentais na metodologia *DevOps*, pode ser reutilizada para garantir a segurança e confiabilidade da transferência de modelos ao utilizar a técnica de TL.

Diante desses resultados, torna-se evidente que a evolução e otimização contínuas das técnicas de aprendizado são essenciais para fortalecer a segurança cibernética, especialmente em um mundo cada vez mais digitalizado e interconectado. Dessa forma, este estudo incentiva a comunidade acadêmica e os profissionais da área a aprofundarem suas pesquisas e práticas, ampliando o uso do FL na mitigação de ameaças e na proteção de dados sensíveis.

Por fim, recomenda-se que futuras pesquisas explorem a utilização de métodos alternativos de agregação, como os algoritmos *FedAvgM*, *FedAdam* e *FedAdagrad*, além de aprimoramentos na técnica FTL. Observou-se que não há *frameworks* de FL que ofereçam essa funcionalidade nativamente, o que evidencia um campo pouco explorado e repleto de

oportunidades para a investigação e desenvolvimento de modelos mais eficientes e adaptáveis para dispositivos IoT e *Edge Computing*.

Cabe ressaltar que um trabalho intitulado "Detection of Obfuscation Malware: A Federated Transfer Learning-based Approach with Hybrid Neural Networks" foi submetido à REVISTA IEEE AMÉRICA LATINA (IEEE Latin America Transactions), Qualis Capes A4, o artigo encontra-se anexo.

Referências

ANDERSON, J. P. Computer security threat monitoring and surveillance. *Technical Report, James P. Anderson Company*, 1980.

AOUEDI, O. et al. Intrusion detection for softwarized networks with semi-supervised federated learning. In: IEEE. *ICC 2022-IEEE International Conference on Communications*. [S.l.], 2022. p. 5244–5249.

BANALA, S. Devops essentials: Key practices for continuous integration and continuous delivery. *International Numeric Journal of Machine Learning and Robots*, v. 8, n. 8, p. 1–14, 2024.

BERTOLI, G. de C.; JÚNIOR, L. A. P.; SAOTOME, O. Improving detection of scanning attacks on heterogeneous networks with federated learning. *ACM SIGMETRICS Performance Evaluation Review*, ACM New York, NY, USA, v. 49, n. 4, p. 118–123, 2022.

BEUTEL, D. J. et al. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.

BIOLCHINI, J. et al. Systematic review in software engineering. *System engineering and computer science department COPPE/UFRJ, Technical Report ES*, v. 679, n. 05, p. 45, 2005.

BOZINOVSKI, S.; FULGOSI, A. The influence of pattern similarity and transfer learning upon training of a base perceptron b2. In: *Proceedings of Symposium Informatica*. [S.l.: s.n.], 1976. v. 3, p. 121–126.

CAMARGO, L. F. de et al. Métodos de aprendizado de máquina adversariais na detecçao de anomalias em redes de computadores. In: SBC. *Anais do XXI Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*. [S.l.], 2021. p. 169–182.

CAO, K. et al. An overview on edge computing research. *IEEE access*, IEEE, v. 8, p. 85714–85728, 2020.

CATILLO, M. et al. Transferability of machine learning models learned from public intrusion detection datasets: the cicids2017 case study. *Software Quality Journal*, Springer, p. 1–27, 2022.

CEVALLOS-SALAS, D. et al. Obfuscated privacy malware classifiers based on memory dumping analysis. *IEEE Access*, IEEE, 2024.

CHAUHAN, V. K. et al. A brief review of hypernetworks in deep learning. *Artificial Intelligence Review*, Springer, v. 57, n. 9, p. 250, 2024.

CHEN, S. et al. An efficient intrusion detection method based on federated transfer learning. In: IEEE. *2024 3rd International Conference on Artificial Intelligence and Computer Information Technology (AICIT)*. [S.l.], 2024. p. 1–4.

CHEN, Z. et al. Intrusion detection for wireless edge networks based on federated learning. *IEEE Access*, IEEE, v. 8, p. 217463–217472, 2020.

CHENG, Y. et al. Federated transfer learning with client selection for intrusion detection in mobile edge computing. *IEEE Communications Letters*, IEEE, v. 26, n. 3, p. 552–556, 2022.

DEB, S. K. et al. Wi-fi optimization using parabolic reflector and blocking materials in intrusion detection systems. In: SPRINGER. *Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2018, Volume 3*. [S.l.], 2019. p. 761–771.

DENNING, D. E. An intrusion-detection model. *IEEE Transactions on software engineering*, IEEE, n. 2, p. 222–232, 1987.

DHILLON, H.; HAQUE, A. Towards network traffic monitoring using deep transfer learning. In: IEEE. *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. [S.l.], 2020. p. 1089–1096.

ESTUDOS, R. e. T. d. I. d. S. n. B. Centro de. Incidentes notificados ao cert.br. 2025.

FAN, Y. et al. Iotdefender: A federated transfer learning intrusion detection framework for 5g iot. In: IEEE. *2020 IEEE 14th international conference on big data science and engineering (BigDataSE)*. [S.l.], 2020. p. 88–95.

Flower Authors. *Flower: A Friendly Federated Learning Framework*. 2024. [Accessed: 2024-12-02]. Disponível em: <https://flower.dev>.

FUNG, G.; SANDILYA, S.; RAO, R. B. Rule extraction from linear support vector machines. In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. [S.l.: s.n.], 2005. p. 32–40.

GAO, Y. et al. Adaptive-hmd: Accurate and cost-efficient machine learning-driven malware detection using microarchitectural events. In: IEEE. *2021 IEEE 27th International Symposium on On-Line Testing and Robust System Design (IOLTS)*. [S.l.], 2021. p. 1–7.

GAZEAU, V.; GUPTA, K.; AN, M. K. Advancements of machine learning in malware and intrusion detections. In: IEEE. *2024 International Conference on Computer, Information and Telecommunication Systems (CITS)*. [S.l.], 2024. p. 1–7.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep learning*. [S.l.]: MIT press, 2016.

GU, J.; LU, S. *An effective intrusion detection approach using SVM with naïve Bayes feature embedding*. [S.l.]: Elsevier, 2021. 102158 p.

HE, Z. et al. Deep neural network and transfer learning for accurate hardware-based zero-day malware detection. In: *Proceedings of the Great Lakes Symposium on VLSI 2022*. [S.l.: s.n.], 2022. p. 27–32.

HUSSAIN, F. et al. Iot dos and ddos attack detection using resnet. In: IEEE. *2020 IEEE 23rd International Multitopic Conference (INMIC)*. [S.l.], 2020. p. 1–6.

IBM, P. I. S. Relatório de custo da violação de dados de 2022. 2022.

IBM Research. *IBM Federated Learning Community Edition*. 2024. [Online] - [Accessed: 2024-12-04]. Disponível em: <https://ibmfl.mybluemix.net>.

JI, X.; ZHANG, H.; MA, X. A novel method of intrusion detection based on federated transfer learning and convolutional neural network. In: IEEE. *2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*. [S.l.], 2022. v. 10, p. 338–343.

KARIMIREDDY, S. P. et al. Federated learning showdown: The comparative analysis of federated learning frameworks. In: IEEE. *2023 Eighth International Conference on Fog and Mobile Edge Computing (FMEC)*. [S.l.], 2023. p. 224–231.

KITCHENHAM, B.; CHARTERS, S. Guidelines for performing systematic literature reviews in software engineering. Citeseer, 2007.

KOMISAREK, M. et al. How to effectively collect and process network data for intrusion detection? *Entropy*, Multidisciplinary Digital Publishing Institute, v. 23, n. 11, p. 1532, 2021.

LALOUANI, W.; YOUNIS, M. Robust distributed intrusion detection system for edge of things. In: IEEE. *2021 IEEE Global Communications Conference (GLOBECOM)*. [S.l.], 2021. p. 01–06.

LAZZARINI, R.; TIANFIELD, H.; CHARISSIS, V. Federated learning for iot intrusion detection. *Ai*, MDPI, v. 4, n. 3, p. 509–530, 2023.

LEARNING, F. Collaborative machine learning without centralized training data. *Publication date: Thursday, April*, v. 6, 2017.

LECUN, Y.; BENGIO, Y. et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, Cambridge, MA USA, v. 3361, n. 10, p. 1995, 1995.

LEONARDO, M. M. et al. Deep feature-based classifiers for fruit fly identification (diptera: Tephritidae). In: IEEE. *2018 31st SIBGRAPI conference on graphics, patterns and images (SIBGRAPI)*. [S.l.], 2018. p. 41–47.

LI, K. et al. Distributed network intrusion detection system in satellite-terrestrial integrated networks using federated learning. *IEEE Access*, IEEE, v. 8, p. 214852–214865, 2020.

LIU, Y. et al. A secure federated transfer learning framework. *IEEE Intelligent Systems*, IEEE, v. 35, n. 4, p. 70–82, 2020.

LUCAS, T. J. et al. Stacking-based committees para detecçao de ataques em redes de computadores-uma abordagem por exaustao. In: SBC. *Anais do XXXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. [S.l.], 2021. p. 644–657.

MASUM, M.; SHAHRIAR, H. Tl-nid: Deep neural network with transfer learning for network intrusion detection. In: IEEE. *2020 15th International Conference for Internet Technology and Secured Transactions (ICITST)*. [S.l.], 2020. p. 1–7.

MCHUGH, J. Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Transactions on Information and System Security (TISSEC)*, ACM New York, NY, USA, v. 3, n. 4, p. 262–294, 2000.

MCMAHAN, B. et al. Communication-efficient learning of deep networks from decentralized data. In: PMLR. *Artificial intelligence and statistics*. [S.l.], 2017. p. 1273–1282.

MCMAHAN, H. B. et al. Communication-efficient learning of deep networks from decentralized data. *arXiv e-prints*, p. arXiv–1602, 2016.

MIRZAEE, P. H. et al. Fids: A federated intrusion detection system for 5g smart metering network. In: IEEE. *2021 17th International Conference on Mobility, Sensing and Networking (MSN)*. [S.l.], 2021. p. 215–222.

MOUSTAFA, N. New generations of internet of things datasets for cybersecurity applications based machine learning: Ton_iot datasets. In: *Proceedings of the eResearch Australasia Conference, Brisbane, Australia*. [S.l.: s.n.], 2019. p. 21–25.

MOUSTAFA, N. et al. Federated ton_iot windows datasets for evaluating ai-based security applications. In: IEEE. *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. [S.l.], 2020. p. 848–855.

NETO, H. N.; MATTOS, D. M.; FERNANDES, N. C. Privacidade do usuário em aprendizado colaborativo: Federated learning, da teoria à prática. *Sociedade Brasileira de Computação*, 2020.

NGUYEN, T. D. et al. Dïot: A federated self-learning anomaly detection system for iot. In: IEEE. *2019 IEEE 39th International conference on distributed computing systems (ICDCS)*. [S.l.], 2019. p. 756–767.

NVIDIA. *NVFlare: NVIDIA Federated Learning Application Runtime Environment*. 2024. [Online] - [Accessed: 2024-12-01]. Disponível em: <https://nvflare.readthedocs.io>.

OpenMined. *PySyft: A Library for Secure and Private Machine Learning*. 2024. [Online] - [Accessed: 2024-11-13]. Disponível em: <https://github.com/OpenMined/PySyft>.

OSKEN, S. et al. Intrusion detection systems with deep learning: A systematic mapping study. In: IEEE. *2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT)*. [S.l.], 2019. p. 1–4.

OTOUM, Y.; NAYAK, A. Signature-over-the-air with transfer learning ids for intelligent connected vehicles (icv). In: IEEE. *2021 IEEE Globecom Workshops (GC Wkshps)*. [S.l.], 2021. p. 1–6.

PAN, S. J.; YANG, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, IEEE, v. 22, n. 10, p. 1345–1359, 2010.

PAWLICKI, M.; KOZIK, R.; CHORAŚ, M. Towards deployment shift inhibition through transfer learning in network intrusion detection. In: *Proceedings of the 17th International Conference on Availability, Reliability and Security*. [S.l.: s.n.], 2022. p. 1–6.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, JMLR. org, v. 12, p. 2825–2830, 2011.

PETROV, D.; HOSPEDALES, T. M. Measuring the transferability of adversarial examples. *arXiv preprint arXiv:1907.06291*, 2019.

POPOOLA, S. I. et al. Federated deep learning for collaborative intrusion detection in heterogeneous networks. In: IEEE. *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*. [S.l.], 2021. p. 1–6.

PRATT, D. D. Andragogy after twenty-five years. *New directions for adult and continuing education*, v. 57, n. 57, p. 15–23, 1993.

RAJESH, L. T. et al. Give and take: Federated transfer learning for industrial iot network intrusion detection. *arXiv e-prints*, p. arXiv–2310, 2023.

SAHA, S.; AHMAD, T. Federated transfer learning: concept and applications. *Intelligenza Artificiale*, IOS Press, v. 15, n. 1, p. 35–44, 2021.

SATYANARAYANAN, M. The emergence of edge computing. *Computer*, IEEE, v. 50, n. 1, p. 30–39, 2017.

SEO, E.; SONG, H. M.; KIM, H. K. Gids: Gan based intrusion detection system for in-vehicle network. In: IEEE. *2018 16th Annual Conference on Privacy, Security and Trust (PST)*. [S.l.], 2018. p. 1–6.

SHARAFALDIN, I.; LASHKARI, A. H.; GHORBANI, A. A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, v. 1, p. 108–116, 2018.

SHARMA, K. et al. Combating fake news: A survey on identification and mitigation techniques. *ACM Transactions on Intelligent Systems and Technology (TIST)*, ACM New York, NY, USA, v. 10, n. 3, p. 1–42, 2019.

SHI, W. et al. Edge computing: Vision and challenges. *IEEE internet of things journal*, IEEE, v. 3, n. 5, p. 637–646, 2016.

SHI, W. et al. Edge computing: state-of-the-art and future directions. *Journal of Computer Research and Development*, v. 56, n. 1, p. 69–89, 2019.

SHI, Y. et al. Experimenting fedml and nvflare for federated tumor segmentation challenge. In: SPRINGER. *International MICCAI Brainlesion Workshop*. [S.l.], 2022. p. 228–240.

SHI, Y. et al. User privacy leakages from federated learning in nilm applications. In: *Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. [S.l.: s.n.], 2021. p. 212–213.

SHOKRI, R.; SHMATIKOV, V. Privacy-preserving deep learning. In: *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. [S.l.: s.n.], 2015. p. 1310–1321.

SINGLA, A.; BERTINO, E.; VERMA, D. Overcoming the lack of labeled data: Training intrusion detection models using transfer learning. In: IEEE. *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*. [S.l.], 2019. p. 69–74.

STERLE, L.; BHUNIA, S. On solarwinds orion platform security breach. In: *2021 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/IOP/SCI)*. [S.l.: s.n.], 2021. p. 636–641.

SUN, Y.; OCHIAI, H.; ESAKI, H. Intrusion detection with segmented federated learning for large-scale multiple lans. In: IEEE. *2020 International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2020. p. 1–8.

TAN, J. et al. A critical look at the current train/test split in machine learning. *arXiv preprint arXiv:2106.04525*, 2021.

TAVALLAEE, M. et al. A detailed analysis of the kdd cup 99 data set. In: IEEE. *2009 IEEE symposium on computational intelligence for security and defense applications*. [S.l.], 2009. p. 1–6.

VINAYAKUMAR, R. et al. Robust intelligent malware detection using deep learning. *IEEE access*, IEEE, v. 7, p. 46717–46738, 2019.

WANG, H. et al. Mitigating cache-based side-channel attacks through randomization: A comprehensive system and architecture level analysis. In: IEEE. *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. [S.l.], 2020. p. 1414–1419.

WANG, K. et al. An efficient intrusion detection method based on federated transfer learning and an extreme learning machine with privacy preservation. *Security and Communication Networks*, Hindawi, v. 2022, 2022.

WU, P.; GUO, H.; BUCKLAND, R. A transfer learning approach for network intrusion detection. In: IEEE. *2019 IEEE 4th international conference on big data analytics (ICBDA)*. [S.l.], 2019. p. 281–285.

XU, Y. et al. Intrusion detection based on fusing deep neural networks and transfer learning. In: SPRINGER. *Digital TV and Wireless Multimedia Communication: 16th International Forum, IFTC 2019, Shanghai, China, September 19–20, 2019, Revised Selected Papers 16*. [S.l.], 2020. p. 212–223.

YANG, L.; SHAMI, A. A transfer learning and optimized cnn based intrusion detection system for internet of vehicles. In: IEEE. *ICC 2022-IEEE International Conference on Communications*. [S.l.], 2022. p. 2774–2779.

YANG, Q. et al. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, ACM New York, NY, USA, v. 10, n. 2, p. 1–19, 2019.

ZHANG, C. et al. A survey on federated learning. *Knowledge-Based Systems*, Elsevier, v. 216, p. 106775, 2021.

ZHANG, J. et al. Federated learning for distributed iiot intrusion detection using transfer approaches. *IEEE Transactions on Industrial Informatics*, IEEE, v. 19, n. 7, p. 8159–8169, 2022.

ZHANG, Y.; YAN, J. Domain-adversarial transfer learning for robust intrusion detection in the smart grid. In: IEEE. *2019 IEEE international conference on communications, control, and computing technologies for smart grids (SmartGridComm)*. [S.l.], 2019. p. 1–6.