Universidade Estadual de Londrina

BLENDA OLIVEIRA MAZETTO

APRENDIZADO FEDERADO PARA DETECÇÃO DE INTRUSÕES

LONDRINA
2023

BLENDA OLIVEIRA MAZETTO

APRENDIZADO FEDERADO PARA DETECÇÃO DE INTRUSÕES

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação do Departamento de Computação da Universidade Estadual de Londrina, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Bruno Bogaz Zarpelão

LONDRINA

2023

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

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

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

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

BLENDA OLIVEIRA MAZETTO

APRENDIZADO FEDERADO PARA DETECÇÃO DE INTRUSÕES

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação do Departamento de Computação da Universidade Estadual de Londrina, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA

Orientador: Prof. Dr. Bruno Bogaz Zarpelão
Universidade Estadual de Londrina

Prof. Dr. Segundo Membro da Banca
Universidade/Instituição do Segundo
Membro da Banca – Sigla instituição

Prof. Dr. Terceiro Membro da Banca
Universidade/Instituição do Terceiro
Membro da Banca – Sigla instituição

Londrina, 03 de maio de 2023.

AGRADECIMENTOS

Agradeço aos professores do Departamento de Computação da Universidade Estadual de Londrina pelos ensinamentos. Em especial, agradeço meu professor orientador Bruno Bogaz Zarpelão pela disponibilidade, e por todo o suporte necessário para o desenvolvimento desse trabalho.

Agradeço aos meus colegas de turma que sempre foram gentis e, em especial, agradeço aos meus amigos, Pedro, Rafael e Laura que me ajudaram a não desistir nos tempos difíceis e enfrentar os desafios da graduação juntos.

Por fim, gostaria de agradecer a minha família que sempre me apoiou, me entendeu e acreditou em mim.

"Não vos amoldeis às estruturas deste mundo, mas transformai-vos pela renovação da mente, a fim de distinguir qual é a vontade de Deus: o que é bom, o que Lhe é agradável, o que é perfeito.
(Bíblia Sagrada, Romanos 12, 2))

MAZETTO, B. O.. **Aprendizado Federado para Detecção de Intrusões**. 2023. 48f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2023.

RESUMO

Nas últimas décadas, a Internet tornou-se cada vez mais uma parte essencial da vida cotidiana e com isso, as intrusões de rede se tornaram um grande problema. Neste cenário, as técnicas de Aprendizado de Máquina (ML - *Machine Learning*) estão desempenhando um papel fundamental em Sistemas de Detecção de Intrusão (IDS - *Intrusion Detection Systems*). As técnicas de ML podem ser usadas para treinar modelos que aprendam a diferenciar padrões de tráfego de rede normal e anormal, permitindo que o IDS identifique atividades maliciosas. Para treinar algoritmos supervisionados de detecção de intrusão, é necessário ter um conjunto de dados que contenha exemplos tanto de tráfego benigno quanto malicioso. No entanto, a grande quantidade de dados necessários para o treinamento levanta preocupações sobre a privacidade das informações utilizadas, pois as amostras de tráfego podem conter informações sensíveis ou confidenciais. Neste contexto, o aprendizado federado foi criado para treinar modelos de aprendizado de máquina em dados distribuídos, sem que esses dados precisem ser centralizados em um único local, desta forma, auxiliando na proteção da privacidade. Para estudar a aplicação de aprendizado federado em IDSs, este trabalho propõe três sistemas que serão avaliados e comparados entre si. O primeiro sistema usa o aprendizado federado e possui três *workers*, cada *worker* é composto por um conjunto de dados emulando uma rede doméstica. O segundo sistema usa um aprendizado local e o terceiro sistema usa um aprendizado centralizado, enviando seus dados para um servidor central que realiza o treinamento, ambos utilizam os mesmos conjuntos de dados citados anteriormente.

Palavras-chave: Preservação de Privacidade. Sistemas de Detecção de Intrusão. Aprendizado Federado. Aprendizado de Máquina.

MAZETTO, B. O.. **Federated Learning for Intrusion Detection**. 2023. 48p. Final Project (Bachelor of Science in Computer Science) – State University of Londrina, Londrina, 2023.

ABSTRACT

In recent decades, the Internet has become an essential part of everyday life, and with this, network intrusions have become a significant problem. In this scenario, Machine Learning (ML) techniques are playing a fundamental role in Intrusion Detection Systems (IDS) by detecting malicious activities. Machine Learning techniques can be used to train models that can learn normal and abnormal network traffic patterns, allowing the IDS to identify malicious activities. To train supervised intrusion detection algorithms, it is necessary to have a dataset that contains examples of both benign and malicious traffic. However, the large amount of data required for supervised intrusion detection algorithm training raises concerns about the privacy of the information used, as traffic samples may contain sensitive or confidential information. In this context, federated learning was created to train machine learning models on distributed data, without centralizing the data in a single location, allowing privacy preservation. To study the application of federated learning in intrusion detection systems, this work proposes three systems that will be evaluated and compared. The first system uses federated learning, has three workers, and consists of three datasets, each emulating a local network. The second system uses local learning, and the third system uses centralized learning, sending its data to a central server that performs the training, both using the same datasets mentioned earlier.

Keywords: Privacy Preserving. Intrusion Detection Systems. Federated Learning. Machine Learning.

LISTA DE ILUSTRAÇÕES

Figura 1 – Exemplo de uma rede neural multicamadas. 20
Figura 2 – Diagrama de blocos Aprendizado Federado. Fonte: [1] 22
Figura 3 – Diagrama de blocos Aprendizado Federado. 28
Figura 4 – Diagrama de blocos Aprendizado Local. 29
Figura 5 – Diagrama de blocos Aprendizado Centralizado. 29

LISTA DE TABELAS

Tabela 1 – Características extraídas do fluxo. 30
Tabela 2 – Relação entre as nove categorias de ataques e os IPs estáticos do *Kali Linux* que estão propagando os ataques. Fonte: [2] 32
Tabela 3 – Dispositivos IoT usados no conjunto de dado CIC-IOT. Fonte: [3] . . . 34
Tabela 4 – Regras de rotulação para IoT-23. Fonte: [4] 36
Tabela 5 – Resultados do sistema que utilizou aprendizado federado. 39
Tabela 6 – Resultados do sistema que utilizou aprendizado local. 41
Tabela 7 – Resultados do sistema que utilizou aprendizado centralizado. 42

LISTA DE ABREVIATURAS E SIGLAS

ML	Machine Learning
IDS	Intrusion Detection Systems
FL	Federated Learning
IoT	Internet of Things
M2M	Machine-to-Machine
IA	Inteligência Artificial
IIoT	Industrial Internet of Things
NN	Neural Network
SGD	Stochastic Gradient Descent
ReLU	Rectified Linear Unit
TP	True Positive
FP	False Positive
TN	True Negative
FN	False negative

SUMÁRIO

1 INTRODUÇÃO 13

2 FUNDAMENTAÇÃO TEÓRICA E ESTADO DA ARTE . . . 15
2.1 Internet das Coisas 15
2.2 Sistemas de Detecção de Intrusão 17
2.3 Aprendizado de Máquina 18
2.3.1 Redes Neurais 19
2.4 Aprendizado Federado 21
2.4.1 Trabalhos Relacionados 24

3 MATERIAIS E MÉTODOS 27
3.1 VISÃO GERAL DO EXPERIMENTO 27
3.2 Conjuntos de Dados 31
3.2.1 TON_IOT 31
3.2.2 CIC IoT dataset 2022 32
3.2.3 IoT-23 35
3.3 Redes Neurais 36
3.4 Implementação 37

4 RESULTADOS 38
4.1 Aprendizado Federado 38
4.2 Aprendizado Local 40
4.3 Aprendizado Centralizado 42
4.4 Discussão 42

5 CONCLUSÃO 44

REFERÊNCIAS 45

1 INTRODUÇÃO

Com o rápido desenvolvimento da tecnologia, a Internet se tornou parte da vida cotidiana e uma ferramenta essencial em diversos campos. Como consequência, as intrusões em redes comerciais e privadas têm sido motivo de preocupação. O aprendizado de máquina vem sendo amplamente utilizado para a detecção de intrusão, por permitir criar sistemas facilmente adaptáveis a diferentes realidades e contextos. Além disso, são diversas as técnicas existentes utilizadas nos Sistemas de Detecção de Intrusão (IDS - *Intrusion Detection Systems*) [5].

Apesar de seu potencial na detecção de intrusões, o uso do aprendizado de máquina pode levantar questões sobre a privacidade das informações usadas em seu treinamento, especialmente por conta da demanda por grandes volumes de dados. Em 2018, por exemplo, foi revelado que o Facebook treinou um modelo de detecção de objetos usando 3,5 bilhões de imagens obtidas do Instagram. O modelo treinado superou os outros modelos existentes demonstrando a importância da quantidade de dados [6]. Esse evento fez com que a privacidade fosse novamente uma preocupação dos usuários da Internet, motivando a comunidade científica a buscar por técnicas de aprendizado de máquina que tenham como característica a preservação de privacidade.

Para detectar ataques em redes e sistemas de computadores, os IDSs precisam coletar, armazenar e analisar uma ampla gama de dados. Esses dados podem conter informações sensíveis como endereço IP, dados de tráfego de rede, logs de eventos, pacotes de rede, entre outros, portanto, comprometendo a privacidade [7].

O Aprendizado Federado (FL - *Federated Learning*) [8] traz a possibilidade de um treinamento descentralizado, onde os dados locais são privados e o objeto compartilhado é um modelo de aprendizado. Durante o treinamento, cada um dos participantes do aprendizado federado começa com um modelo genérico inicial fornecido por um servidor central seguro. Com seus dados locais, cada participante realiza o treinamento do modelo recebido, e após a etapa de treinamento, cada um dos participantes envia seu modelo treinado para o servidor central. Então, o servidor agrega esses modelos gerando um novo modelo de aprendizado. O modelo agregado é enviado para os participantes e repetem-se as etapas citadas anteriormente até que o treinamento esteja completo.

Considerando as três abordagens do aprendizado de máquina, supervisionado, não-supervisionado e por reforço, a literatura apresenta diferentes técnicas usando o aprendizado federado. Dentro do aprendizado supervisionado, [1, 9, 10, 11, 12, 13] trazem modelos de FL aplicados em sistemas de detecção de intrusão. Ainda no âmbito de IDS, temos [14] e [15] que usam a abordagem de aprendizado não supervisionado. Ainda usando a abor-

dagem não supervisionada, [16] utiliza a técnica de clusterização. Já em [17], podemos observar uma técnica de FL usando a abordagem de aprendizado por reforço.

No campo de Internet das Coisas (IoT - *Internet of Things*), o FL vem ganhando importância e diversos trabalhos já foram desenvolvidos com base nesse paradigma [18]. Em [19] é realizada uma investigação das possibilidades trazidas pelo aprendizado federado em relação à detecção de *malwares* em IoT, além de um estudo de questões de segurança inerentes a esse novo paradigma de aprendizado.

Neste trabalho, é estudado o uso do aprendizado federado para a detecção de intrusão em uma tentativa de colaborar para uma maior privacidade dos dados. O objetivo é avaliar o desempenho preditivo do aprendizado federado em comparação com um aprendizado de máquina centralizado em IDSs. Para isso, foram propostos três modelos de detecção de intrusão que serão avaliados e comparados entre si. Para o treinamento dos sistemas, foram utilizados três conjuntos de dados: CIC-IOT, TON-IOT e IOT-23. Os conjuntos de dados escolhidos tiveram seus dados coletados em ambientes com dispositivos IoT. Além disso, todos eles possuem amostras naturais e de ataques.

O primeiro sistema usa o aprendizado federado, possui três *workers* e é composto pelos três conjuntos de dados, cada um emulando uma rede doméstica. O segundo sistema usa um aprendizado local, apresentando um modelo de rede neural individual para cada conjunto de dados e possibilitando a visualização de um treinamento sem colaboração. Por fim, o terceiro sistema teve os três conjuntos de dados unidos em um de forma aleatória, vendo todos os dados diretamente. O último sistema utiliza o aprendizado centralizado, enviando seus dados para um servidor central que realiza o treinamento. Algumas métricas avaliadas foram: F1-score, *precision*, *recall*, número de verdadeiros positivos, falsos positivos, verdadeiros negativos e falsos negativos.

No Capítulo 2, é apresentada a fundamentação teórica e o estado da arte, desenvolvendo conceitos de IoT, Sistemas de Detecção de Intrusão, Aprendizado de Máquina e Aprendizado Federado. O Capítulo 3 explica todos os materiais e métodos usados para se obter os resultados, começando com uma visão geral e posteriormente passando pelos conjuntos de dados utilizados, as redes neurais e a implementação. No Capítulo 4 são apresentados os resultados obtidos pelos três sistemas desenvolvidos, os resultados também são discutidos neste mesmo Capítulo. Por fim, o Capítulo 5 apresenta a conclusão deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA E ESTADO DA ARTE

2.1 Internet das Coisas

A Internet das Coisas é uma rede de objetos físicos, dispositivos, veículos, edifícios e outros itens incorporados com sensores, software e conectividade que permitem que esses objetos coletem e troquem dados [20]. Essa rede de dispositivos conectados pode se comunicar e interagir entre si e com outros sistemas e redes, como a Internet. A IoT permite a integração dos mundos físico e digital, acolhendo novas formas de troca de informações e automação [21].

O conceito de IoT tem suas raízes nos primórdios da Internet e no desenvolvimento da comunicação máquina a máquina (M2M - $Machine - to - Machine$) [22]. O termo "Internet das Coisas" foi usado pela primeira vez por Kevin Ashton, um pioneiro da tecnologia britânica, em 1999. Ele usou a frase para descrever um sistema em que objetos do cotidiano podem ser conectados à Internet e se comunicar uns com os outros [20].

Nos últimos anos, a IoT vem em uma crescente considerável no cenário global. Ela é usada em uma ampla gama de aplicações, desde casas e cidades inteligentes, automação industrial, assistência médica, transporte, infraestrutura civil, entre outros, trazendo praticidade e conforto à vida humana. Existem várias arquiteturas para aplicações de IoT, a escolha depende das necessidades específicas do projeto e dos requisitos de desempenho [23]. Algumas das principais arquiteturas que podem ser encontradas em aplicações de IoT incluem:

- *Cloud computing*: um modelo para permitir o acesso onipresente, conveniente e sob demanda à rede a um conjunto compartilhado de recursos de computação configuráveis, redes, servidores, armazenamento, aplicativos e serviços que podem ser rapidamente provisionados e liberados com o mínimo de esforço de gerenciamento ou interação do provedor de serviços [24].

- *Edge computing*: refere-se às tecnologias que permitem que a computação seja executada na borda da rede, em dados entre a computação de nuvem e os serviços IoT. Basicamente, a ideia é estender a computação em nuvem até a borda da rede com o objetivo de ter a computação próxima às fontes de dados, ou seja, dispositivos IoT [25].

- *Fog computing*: uma plataforma altamente virtualizada que fornece serviços de computação, armazenamento e rede entre dispositivos finais e *Data Centers* tradicio-

nais de computação em nuvem, geralmente, mas não exclusivamente, localizados na borda da rede [26].

Entretanto, esses sistemas apresentam grandes desafios considerando a segurança de redes, sendo que, para esses equipamentos, métodos convencionais de segurança podem ser inadequados [27]. Redes IoT apresentam arquiteturas diferentes das redes tradicionais, além de muitas vezes utilizarem protocolos de rede diferentes [27]. Sendo assim, são apresentadas vulnerabilidades distintas das arquiteturas e protocolos tradicionais. Desta forma, sistemas de detecção de intrusão convencionais são pouco eficientes para sistemas IoT, visto que foram desenvolvidos para proteger vulnerabilidades específicas das redes tradicionais.

Ainda, considerando o crescente uso de dispositivos IoT em atividades sensíveis à segurança e privacidade de dados, ataques nesses sistemas podem resultar em graves consequências [28]. Neste contexto, a segurança de dispositivos IoT se tornou um grande desafio para a comunidade científica. Dentro dos principais tipos de ameaças aos sistemas de IoT, podemos citar:

1. Senhas fracas e fáceis de adivinhar: uso de credenciais facilmente sujeitas a força bruta, publicamente disponíveis ou imutáveis;

2. Serviços de rede inseguros: serviços de rede inseguros executados no próprio dispositivo, especialmente aqueles expostos à Internet, que comprometem a confidencialidade, integridade/autenticidade ou disponibilidade de informações, ou permitem controle remoto não autorizado;

3. Ecossistema inseguro de interfaces: o ecossistema fora do dispositivo permite o comprometimento do dispositivo ou de seus componentes. Exemplo: web insegura, API de back-end, nuvem ou interfaces móveis. Alguns problemas comuns incluem falta de autenticação/autorização e criptografia fraca;

4. Falta de mecanismo de atualização seguro: inclui falta de validação de firmware no dispositivo, falta de entrega segura (não criptografada em trânsito), falta de controle anti-reversão e falta de notificações de alterações de segurança devido a atualizações;

5. Uso de componentes inseguros ou desatualizados: uso de componentes/bibliotecas de softwares obsoletos ou inseguros que podem permitir que o dispositivo seja comprometido. Isso inclui customização insegura de plataformas de sistema operacional e o uso de software de terceiros ou componentes de hardware de uma cadeia de suprimentos comprometida;

6. Proteção de privacidade insuficiente: informações pessoais do usuário armazenadas no dispositivo ou no ecossistema que são usadas de forma insegura, inadequada ou sem permissão;

7. Transferência e armazenamento de dados inseguros: falta de criptografia ou controle de acesso de dados prejudicado, inclusive em repouso, em trânsito ou durante o processamento;

8. Falta de gerenciamento de dispositivos: falta de suporte de segurança em dispositivos implantados na produção, incluindo gerenciamento de ativos, gerenciamento de atualizações, descomissionamento seguro, monitoramento de sistemas e recursos de resposta;

9. Configurações padrão inseguras: dispositivos ou sistemas enviados com configurações padrão inseguras, ou sem a capacidade de tornar o sistema mais seguro, impedindo que os operadores modifiquem as configurações;

10. Falta de proteção física: falta de medidas de proteção física, permitindo que invasores em potencial obtenham informações avançadas que podem ajudar em um futuro ataque remoto ou assumir o controle local do dispositivo.

2.2 Sistemas de Detecção de Intrusão

Um Sistema de Detecção de Intrusão tem como função monitorar diferentes parâmetros de funcionamento do sistema computacional ou rede de comunicações que visa proteger. Seu objetivo é encontrar eventos que possam violar suas regras de segurança, o IDS pode escolher diferentes abordagens em relação a seu posicionamento e ao seu método de detecção [29].

Um NIDS (*Network Intrusion Detection System*) analisa o fluxo de informações que transitam pela rede verificando os pacotes capturados, o sistema visa encontrar padrões comportamentais suspeitos em busca de tentativas de invasão. Neste sistema o IDS é geralmente posicionado em locais estratégicos da rede [30]. Por outro lado, um sistema HIDS (*Host-based Intrusion Detection Systems*) examina ações específicas com base nos hospedeiros, como, por exemplo, os arquivos acessados, aplicativos utilizados ou informações de logs. Neste sistema, o IDS pode estar presente em todos os hospedeiros ou nos mais críticos, sendo cada um responsável pelo próprio monitoramento [30].

Quanto à estratégia de detecção, os IDSs podem ser implementados de duas formas. O primeiro método é a detecção por assinatura (*Signature-based IDS*), no qual o sistema deve conhecer previamente os padrões utilizados em cada tipo de ataque que ele poderá identificar. Esta estratégia permite a rápida identificação de ataques conhecidos. Por outro

lado, ataques que não tenham padrões similares a algum conhecido podem se passar como tráfego normal [31, 32].

A segunda abordagem é a detecção por anomalia (*Anomaly-based IDS*). Neste método, o sistema busca por ações ou comportamentos que fogem do padrão no dispositivo ou tráfego de rede. Quando um comportamento que difere desses padrões ocorre, ele é catalogado como anômalo. Esta estratégia não necessita que o sistema conheça previamente os padrões dos ataques, porém, pode gerar uma quantidade considerável de falsos positivos [33, 32].

Muitos sistemas de detecção de intrusão ainda sofrem com uma alta taxa de falsos alarmes, gerando muitos alertas para situações pouco ameaçadoras, aumentando a carga dos analistas de segurança, o que pode fazer com que ataques seriamente prejudiciais sejam ignorados. Assim, muitos pesquisadores têm se concentrado no desenvolvimento de IDSs com taxas de detecção mais altas e taxas de alarmes falsos reduzidas. Outra limitação dos IDS é que eles têm dificuldade de detectar ataques desconhecidos. Como os ambientes de rede mudam rapidamente, variantes de ataque e novos ataques surgem constantemente. Assim, é necessário desenvolver IDSs que possam detectar ataques desconhecidos. Para resolver os problemas anteriores, os pesquisadores começaram a se concentrar na construção de IDSs usando métodos de aprendizado de máquina [34].

2.3 Aprendizado de Máquina

O Aprendizado de máquina é um ramo da Inteligência Artificial (IA) projetado para emular a inteligência humana aprendendo com o ambiente ao redor por meio da experiência [35]. Por meio do uso de métodos estatísticos, os algoritmos são treinados para fazer classificações ou previsões. O processo de aprendizado começa com a coleta e preparação dos dados, esses dados servirão como entrada para o modelo de aprendizado. O modelo é então treinado para reconhecer padrões nos dados e extrair informações relevantes para a tarefa que deseja realizar. Durante o treinamento, o modelo ajusta seus parâmetros para minimizar um erro de previsão entre os dados de entrada e as saídas desejadas. Uma vez que o modelo está treinado, ele pode ser usado para fazer previsões ou classificações com novos dados de entrada. Os métodos de aprendizado podem ser classificados em supervisionado, não supervisionado, semi-supervisionado e por reforço [36] [37].

- Aprendizado supervisionado: é uma abordagem de aprendizado de máquina no qual o modelo é treinado em um conjunto de dados rotulados. O conjunto de dados consiste em exemplos de entrada (também conhecidos como recursos) e as respectivas saídas (também conhecidas como rótulos ou etiquetas). O objetivo do modelo é aprender a mapear as entradas para as saídas corretas com base nos exemplos de

treinamento. Em outras palavras, o modelo aprende a fazer previsões a partir de novos dados, usando o conhecimento que adquiriu durante o treinamento. Esse tipo de aprendizado é amplamente utilizado em tarefas como classificação e regressão.

- Aprendizado não supervisionado: é uma abordagem de aprendizado de máquina onde o modelo é treinado em um conjunto de dados não rotulados. O objetivo é encontrar padrões ou estruturas ocultas nos dados, sem a ajuda de rótulos ou etiquetas. Em outras palavras, o modelo aprende a agrupar ou segmentar os dados com base em suas características comuns. Esse tipo de aprendizado é útil em tarefas como *clustering*, redução de dimensionalidade e análise de anomalias.

- Aprendizado semi-supervisionado: é uma abordagem de aprendizado de máquina que combina elementos do aprendizado supervisionado e não supervisionado. O modelo é treinado em um conjunto de dados que contém exemplos rotulados e não rotulados. O objetivo é usar os exemplos rotulados para orientar o aprendizado do modelo nos dados não rotulados, para melhorar sua capacidade de generalização. Esse tipo de aprendizado é útil em tarefas em que rotular grandes quantidades de dados pode ser caro ou demorado.

- Aprendizado por reforço: é uma abordagem de aprendizado de máquina onde o modelo é treinado para tomar decisões em um ambiente dinâmico, interagindo com ele por meio de ações. O objetivo do modelo é maximizar uma recompensa (ou minimizar uma penalidade) recebida após cada ação. Em outras palavras, o modelo aprende a realizar ações que levam a um resultado positivo, com base no feedback que recebe do ambiente. Esse tipo de aprendizado é útil em tarefas como jogos, robótica e otimização de sistemas.

2.3.1 Redes Neurais

As Redes Neurais (NN - Neural Networks) são um tipo de algoritmo de aprendizado de máquina inspirado na estrutura e funcionamento do cérebro humano. Enquanto as abordagens tradicionais de computação utilizam séries de blocos para executar as tarefas, as redes neurais são uma coleção de nós interconectados, ou "neurônios", projetados para processar e analisar grandes quantidades de dados [38].

A história das redes neurais remonta às décadas de 1940 e 1950, quando os pesquisadores começaram a experimentar redes neurais artificiais como uma forma de modelar e simular o cérebro humano. Um dos primeiros pioneiros neste campo foi Warren McCulloch, um neurocientista que, junto do matemático Walter Pitts, propôs a ideia de um neurônio de "lógica de limiar" que poderia ser usado para simular o comportamento de neurônios reais [39].

Em 1975, Kunihiko Fukushima apresentou pela primeira vez o conceito de rede neural multicamadas [40]. Mas não foi até as décadas de 1980 e 1990 que as redes neurais começaram a ganhar força significativa. A Figura 1 apresenta um exemplo de uma rede neural multicamadas com uma camada oculta.

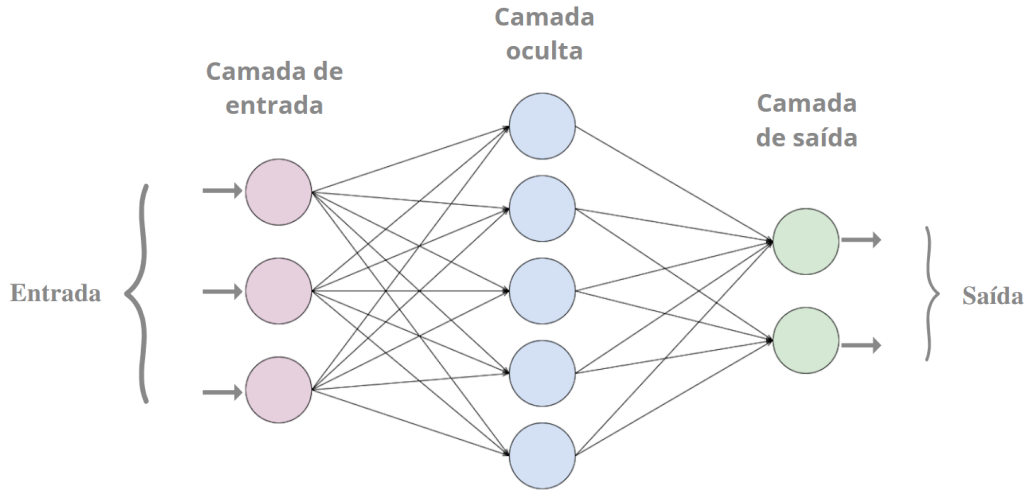

Figura 1 – Exemplo de uma rede neural multicamadas.

A primeira camada de uma rede neural é conhecida como camada de entrada, ela recebe os dados vindos do conjunto de dados que a rede irá analisar. Esses dados passam para as próxima camada, conhecida como camada oculta, onde são processados e analisados. Uma rede neural pode ter várias camadas ocultas. É nas camadas ocultas que ocorre a maior parte da computação, pois elas são compostas por vários neurônios conectados uns aos outros. Finalmente, a camada de saída é onde se produz o resultado final ou previsão.

A rede neural é treinada usando um algoritmo de aprendizado supervisionado, este algoritmo ajusta os pesos e vieses dos neurônios para minimizar o erro entre as saídas previstas e as saídas desejadas para um determinado conjunto de dados. O processo de treinamento envolve várias épocas (iterações) e *batches* (lotes) de dados. Os *batche*s são conjuntos de dados de entrada usados para treinar a rede neural em cada época, dividir os dados em *batches* permite que o treinamento seja realizado em partes menores e mais facilmente gerenciáveis, o que pode ser mais eficiente em termos de memória e computação.

Os pesos e vieses em uma rede neural são os parâmetros ajustáveis que controlam a força e direção das conexões entre os neurônios. Eles são ajustados durante o processo de treinamento para minimizar o erro de previsão. O objetivo do treinamento é encontrar os pesos e vieses ideais que produzem a melhor saída para um determinado conjunto de entradas. Outro ponto importante em uma rede neural são os seus hiperparâmetros, a taxa de aprendizado é um hiperparâmetro que controla o tamanho dos ajustes efetuados nos

pesos e vieses durante o treinamento. Uma taxa de aprendizado maior pode levar a ajustes maiores nos parâmetros, mas pode fazer com que a rede oscile em torno do mínimo local. Uma taxa de aprendizado menor pode levar mais tempo para convergir para o mínimo local, mas pode produzir resultados mais estáveis [41]. O processo de treinamento segue as seguintes etapas:

1. Inicialização dos pesos e vieses: os pesos e bias são inicializados com valores aleatórios pequenos para permitir que o processo de treinamento comece;
2. Propagação para frente (*forward propagation*): a rede neural recebe um conjunto de dados de entrada e processa-o através das camadas de neurônios para gerar uma saída prevista. Durante essa etapa, as entradas são multiplicadas pelos pesos e somadas com os bias de cada neurônio, e então são passadas através de uma função de ativação, que determina a saída do neurônio;
3. Cálculo do erro: o erro é calculado comparando a saída gerada pela rede com a saída desejada para os dados de entrada;
4. Ajuste dos pesos e vieses: a taxa de aprendizado é usada para ajustar os pesos e vieses de cada neurônio na rede, com o objetivo de minimizar o erro;
5. Repetição do processo: os passos 2 a 4 são repetidos para cada *batch* de dados de treinamento durante n épocas.

Com o advento de computadores mais poderosos e grandes conjuntos de dados, os pesquisadores foram capazes de criar modelos mais precisos e sofisticados que poderiam ser aplicados a uma ampla gama de aplicações, incluindo reconhecimento de imagem, reconhecimento de fala e processamento de linguagem natural.

2.4 Aprendizado Federado

No aprendizado de máquina, o processo de treinamento costuma demandar uma grandes quantidades de dados. Tratando-se de um sistema de detecção de intrusão, existe a necessidade de coletar, armazenar e analisar um amplo volume de dados. Esses dados referentes a usuários e organizações podem conter informações privadas que não deveriam ser compartilhadas sem nenhum tipo de proteção, levantando preocupações em relação à privacidade. O aprendizado federado permite realizar aprendizado de máquina de forma colaborativa, sem que haja o compartilhamento de dados, apenas do modelo de aprendizado. Assim, todos os dados de treino permanecem nos dispositivos locais e nenhuma atualização individual é armazenada no servidor [42].

Existem vários termos para se referir aos participantes do aprendizado federado, como: nodes, dispositivos, clientes, entre outros. No entanto, o termo comumente utilizado é "*workers*".

A Figura 2 apresenta um diagrama de blocos FL. Onde w_{j+1} representa os parâmetros do modelo de aprendizado que foram carregados para o servidor, enquanto w'_{j+1} representa os parâmetros que foram agregados pelo servidor e agora estão retornando para o *worker*. Inicialmente, os *workers* recebem modelo global compartilhado e atualizam seus modelos locais. Em seguida, cada *worker* treina seu modelo usando os seus dados de treinamento locais. Após o treinamento, os parâmetros desses modelos serão carregados para o servidor central para agregação. Assim, com base nos parâmetros agregados, o modelo global é atualizado para a próxima iteração do aprendizado colaborativo.

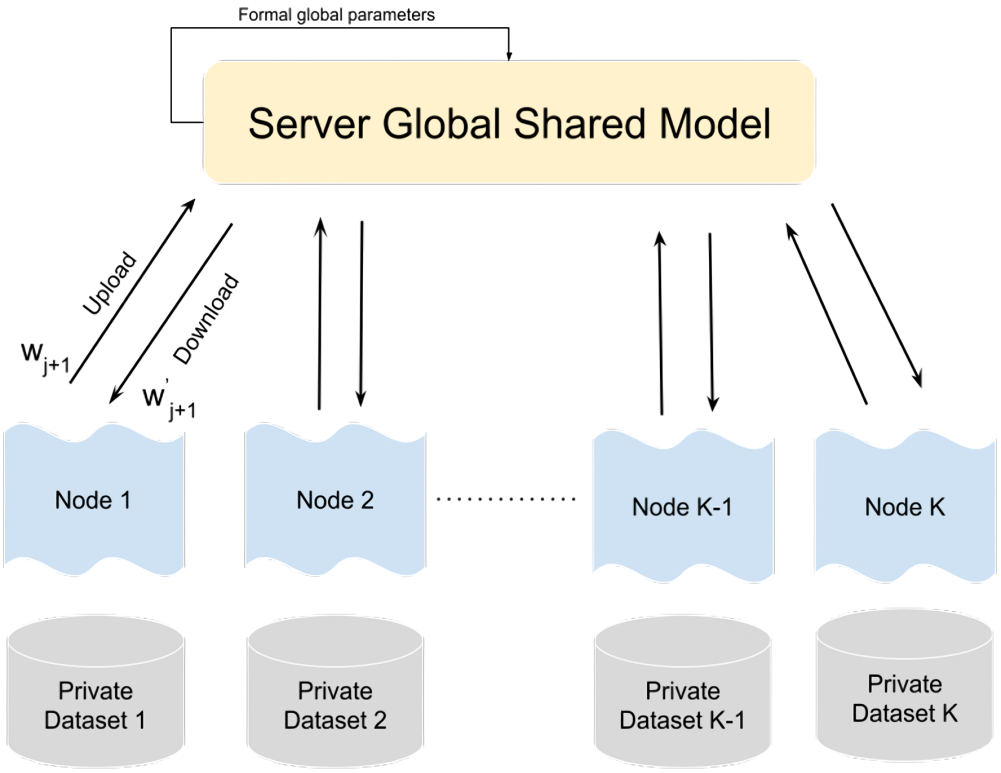

Figura 2 – Diagrama de blocos Aprendizado Federado. Fonte: [1]

O funcionamento do aprendizado federado pode ser descrito em algumas etapas, ilustradas na Figura 2:

1. Os dispositivos participantes do treinamento, denominados Node 1, Node 2 ... Node k, recebem o modelo do servidor central;
2. Cada um dos Nodes treina o seu modelo recebido com seu conjunto de dados privado;
3. Todos os Nodes enviam o modelo treinado para o servidor central;

4. O servidor central agrega os modelos recebidos em único modelo;
5. O servidor central envia o modelo agregado para todos os Nodes.

A inferência no aprendizado federado funciona de maneira semelhante à fase de treinamento. Os dispositivos recebem o modelo global e a partir disso, realizam a inferência sobre dados que eles obtêm localmente. Gerando, assim, as previsões e classificando os novos dados. Um passo importante durante o aprendizado federado é a agregação, ela pode ser realizada de diferentes formas, a seguir serão apresentados alguns exemplos de diferentes algoritmos de agregação.

- FedAvg: combina a *Stochastic gradient descent* (SGD) local de cada *workers* em um servidor que executa a média dos modelos [43].
- FedProx: o objetivo do FedProx é minimizar a função de perda (função de erro) do modelo de aprendizado de máquina, enquanto, ao mesmo tempo, limita o efeito da variação nos dados de treinamento entre as diferentes máquinas. Isso é importante porque, em sistemas distribuídos, cada dispositivo ou máquina pode ter um conjunto diferente de dados de treinamento, o que pode afetar a qualidade do modelo final. O FedProx visa equilibrar a contribuição de cada máquina ou dispositivo na atualização dos parâmetros do modelo, evitando assim o viés em direção a um subconjunto específico de dados [44].
- Fed+: essa abordagem combina várias técnicas visando melhorar a eficiência e a privacidade do aprendizado federado, incluindo [45]:
  - Compressão de modelo: para reduzir a quantidade de dados transmitidos entre os dispositivos e o servidor central.
  - Seleção de modelo: em vez de treinar o modelo em todos os dispositivos, o Fed+ usa uma seleção de modelo baseada em critérios, na qual apenas os dispositivos que atendem certos critérios de desempenho são selecionados para treinar o modelo.
  - Aprendizado adaptativo: usada para ajustar a taxa de aprendizado do modelo com base no desempenho em cada dispositivo, o que pode ajudar a evitar problemas de divergência do modelo.
  - Privacidade diferencial: para proteger a privacidade dos dados dos usuários durante o treinamento do modelo.
- FedOpt: a ideia principal por trás do FedOpt é adaptar o algoritmo de otimização *Stochastic gradient descent* (SGD) para o cenário do aprendizado federado. Em vez de simplesmente combinar os parâmetros de cada dispositivo, o FedOpt usa uma média ponderada dos parâmetros, na qual os dispositivos com melhor desempenho têm

maior peso. Além disso, o FedOpt usa uma técnica de compressão de modelo para reduzir a quantidade de dados que precisam ser transmitidos entre os dispositivos e o servidor central, o que ajuda a reduzir a sobrecarga de comunicação [46].

- FedAdam: em geral, o FedAdam funciona de maneira semelhante ao FedOpt. No entanto, em vez de usar SGD, o FedAdam usa o algoritmo Adam para otimizar o modelo em cada dispositivo local e combinar os parâmetros atualizados no servidor central. O FedAdam usa uma média ponderada dos gradientes locais de cada dispositivo para atualizar os parâmetros do modelo global e usa uma taxa de aprendizado adaptativa para ajustar a taxa de aprendizado do modelo com base no desempenho do dispositivo. Além disso, o FedAdam também pode incorporar técnicas de privacidade diferencial para proteger a privacidade dos dados do usuário durante o treinamento do modelo [46].

- FedAdagrad: o método Adagrad é um algoritmo de otimização que ajusta a taxa de aprendizado de cada parâmetro do modelo com base em suas atualizações de gradiente anteriores. Ele é projetado para lidar com gradientes esparsos e dados não estacionários, com a vantagem de que os parâmetros que são atualizados com frequência recebem uma taxa de aprendizado menor, enquanto aqueles que são atualizados raramente recebem uma taxa de aprendizado maior [46].

- FedYogi: o Yogi é um algoritmo de otimização de gradiente que combina o Adagrad com o RMSprop, com o objetivo de lidar com problemas de gradientes esparsos e não estacionários, além de evitar a instabilidade numérica que pode ocorrer em alguns casos com o Adagrad. O Yogi adapta a taxa de aprendizado para cada parâmetro, mas também adiciona um termo de *momentum* para lidar com gradientes oscilantes [46].

2.4.1 Trabalhos Relacionados

Considerando as três abordagens do aprendizado de máquina, supervisionado, não supervisionado e por reforço, a literatura apresenta diferentes técnicas usando o aprendizado federado.

Dentro do aprendizado supervisionado, Shingi et al. [1] e Sun et al. [13] propõem o Aprendizado Federado Segmentado (Segmented-FL). Em cada rodada, alguns *workers* são selecionados para realizar o treinamento do modelo em seus dados privados, a partir disso, eles carregam os parâmetros de seu modelo para o modelo global. Assim, os parâmetros de vários modelos locais são agregados para atualizar o modelo global. Caso o *worker* não tenha resultados bons em seu grupo, ocorre uma segmentação criando um novo grupo com os *workers* que não se saíram bem no grupo anterior.

Chen et al. [9], traz um modelo de detecção de intrusão baseado em aprendizado federado chamado FedAGRU que se adapta a *wireless edge networks*. O FedAGRU, usa a ideia de um mecanismo de atenção para calcular a importância dos parâmetros do modelo carregado. O critério de importância é baseado na melhoria do desempenho de classificação do modelo global, assim os *workers* são classificados de acordo com sua importância.

O trabalho de Al-Marri et al. [10] propõe uma integração entre FL e *Mimic Learning*, utilizando as vantagens de ambas as técnicas. Foi desenvolvido uma nova técnica de IDS chamada *Federated Mimic Learning*, aproveitando o FL e o *Mimic Learning* para minimizar a possibilidade de obter dados confidenciais e contra-ataques de engenharia reversa no modelo.

Utilizando a abordagem de aprendizado não supervisionado, Preuveneers et al. [14] aborda o aprendizado federado profundo baseado em blockchain. A combinação de aprendizado federado e blockchain neste trabalho visa solucionar dois problemas: potencial envenenamento dos conjuntos de dados e ataques de tempo e escala. A propriedade de não repúdio de blockchains aumenta a confiabilidade da solução de aprendizado federado, pois a cada atualização de modelo, atualizações locais de peso ou gradiente para aprendizado profundo são associadas criptograficamente a um indivíduo. Ainda usando a abordagem de aprendizado não supervisionado, Briggs et al. [16] utiliza a técnica de clusterização, agrupando os *workers* por similaridade com base em suas atualizações no modelo global conjunto após um determinado número de rodadas de comunicação.

Zhao et al. [15] propõe uma *multi-task deep neural network* em aprendizado federado (MT-DNN-FL) para detectar anomalias de rede e analisar o tráfego de rede. No MT-DNN-FL, os participantes não compartilham seus dados de treinamento com terceiros, o que pode impedir que os dados de treinamento sejam explorados por invasores. O MT-DNN-FL pode executar simultaneamente várias tarefas, tarefa de detecção de anomalias de rede, tarefa de reconhecimento de tráfego e tarefa de classificação de tráfego. Comparado com o uso de vários métodos de aprendizado de tarefa única, o MT-DNN-FL pode economizar tempo de treinamento.

O aprendizado federado, assim como outras técnicas de aprendizado de máquina, pode estar vulnerável a ataques, incluindo a engenharia reversa [47]. Esse tipo de ataque pode ocorrer quando um dispositivo malicioso intercepta os modelos, permitindo que o atacante obtenha informações sensíveis e comprometa a privacidade dos usuários.

O trabalho de Li et al. [48] traz a criação de um novo modelo de detecção de intrusão baseado em aprendizado profundo para CPSs industriais, usando CNN e GRU. Este modelo é altamente eficaz na detecção de vários tipos de ameaças cibernéticas contra CPSs industriais, como negação de serviço (DoS), reconhecimento, injeção de resposta e ataques de injeção de comando. É desenvolvida uma estrutura de aprendizagem federada que, por um lado, permite a construção de um modelo abrangente de detecção de intrusão,

aproveitando os recursos de dados de vários proprietários de CPS industriais (no mesmo domínio). Por outro lado, este quadro oferece suporte ao processamento de dados nas próprias instalações de cada CPS industrial, permitindo a preservação efetiva da privacidade dos recursos de dados. Ainda neste trabalho, foi elaborado um protocolo de comunicação seguro baseado em criptossistema de chave pública Paillier para a estrutura de aprendizado federado desenvolvida, pelo qual a segurança e a privacidade dos parâmetros do modelo através do processo de treinamento podem ser bem preservadas.

No campo de IoT, o FL vem ganhando importância e diversos trabalhos já foram desenvolvidos com base nesse paradigma. Nguyen et al. [18] trabalha com um sistema distribuído de autoaprendizagem para monitoramento de segurança de dispositivos IoT, aplicando uma abordagem de aprendizado federado para agregar perfis de detecção de anomalias para detecção de intrusão. Rey et al. [19] utiliza uma estrutura de segurança que usa FL para detectar, preservando a privacidade, ataques cibernéticos que afetam dispositivos IoT. A estrutura proposta abrange abordagens de detecção e classificação de anomalias usando arquiteturas de redes neurais perceptron multicamada e autoencoder.

3 MATERIAIS E MÉTODOS

Este trabalho visa avaliar o desempenho preditivo do aprendizado federado em comparação com um aprendizado de máquina centralizado em IDSs. Para isso, foram propostos três modelos de detecção de intrusão que serão avaliados e comparados entre si. Para alcançar esse objetivo, foram realizadas as seguintes etapas. O primeiro passo foi a busca por conjuntos de dados. A escassez de um conjunto de dados próprio para aprendizado federado em IDS levou a escolha de três conjuntos de dados diferentes, sendo eles: TON_IOT [2], CIC IoT dataset 2022 [49] e IoT-23 [4] que serão detalhados na seção 3.2. O propósito dessa decisão foi implementar um aprendizado federado onde cada conjunto de dados simulasse uma rede doméstica diferente administrada pelo mesmo provedor.

Posteriormente, iniciou-se a implementação de três sistemas, um sistema utilizando aprendizado federado, utilizando um aprendizado local e um utilizando aprendizado centralizado, as implementações serão explicadas na seção 3.4. Todos os sistemas contaram com redes neurais. Para definir a topologia das redes neurais, foram conduzidas uma série de experimentos com diferentes possibilidades que podem ser vistos na seção 3.3, e, assim, os sistemas foram finalizados. Por fim, os resultados foram coletados e analisados, eles podem ser encontrados no capítulo 4.

Os parâmetros avaliados incluem o número de amostras classificadas como verdadeiro positivo, falso positivo, verdadeiro negativo e falso negativo. A partir das métricas citadas acima, foram avaliadas os seguintes: F1 score, que pode ser vista na Equação 3.1, precision na Equação 3.2 e recall na Equação 3.3.

$$F1 = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{3.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

3.1 VISÃO GERAL DO EXPERIMENTO

Neste trabalho foram implementados três modelos de detecção de intrusão, um com aprendizado federado, um com aprendizado local e um com aprendizado centralizado.

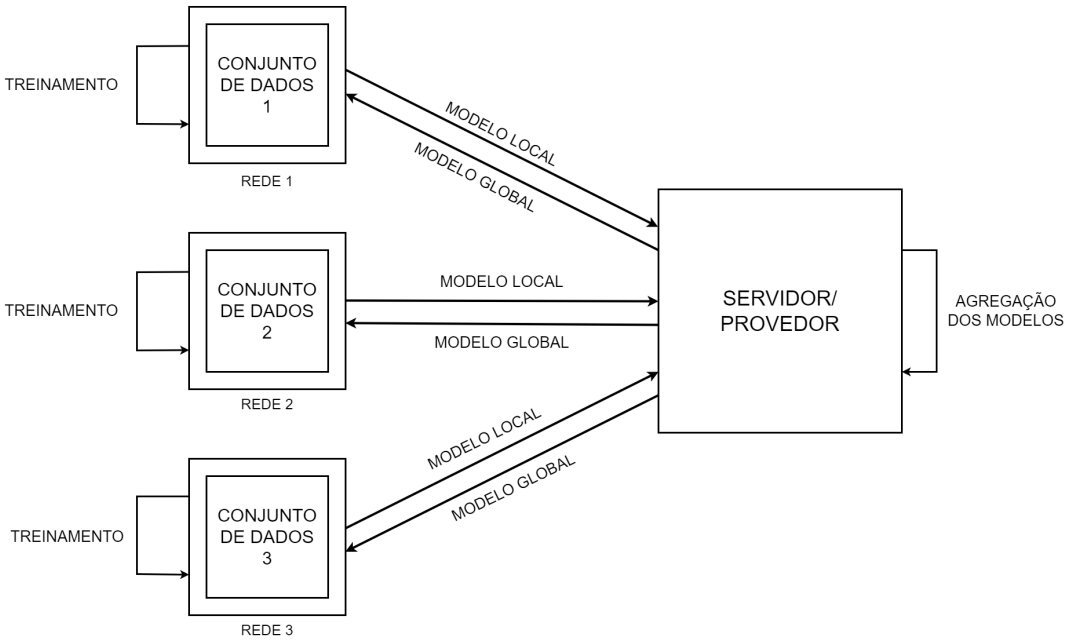

Figura 3 – Diagrama de blocos Aprendizado Federado.

No cenário do aprendizado federado, ilustrado na Figura 3, buscamos simular três *workers*. Cada um emula uma rede doméstica, trabalhando apenas com seus respectivos dados. Sendo assim, cada *worker*, foi composto por amostras vindas de um conjunto de dados diferente.

O treinamento desse sistema é realizado da seguinte forma: primeiramente, um modelo inicial genérico da rede neural é criado pelo servidor central e enviado para cada um dos três *workers*. Cada *worker* usa os seus dados locais para realizar o treinamento, gerando assim três modelos diferentes. Esses modelos são enviados para o servidor central. Ao receber de volta os modelos, o servidor central agrega-os, transformando-os em um único modelo. A agregação é baseada na média dos pesos presentes nos modelos locais. O novo modelo agregado é novamente enviado para os *workers* e esse processo é repetido até o final do treinamento.

A inferência é individual de cada *worker*, ou seja, cada *worker*, utilizando o último modelo que recebeu do servidor central, classifica seus exemplos de teste, sem ter que enviar estes dados para outros elementos na rede.

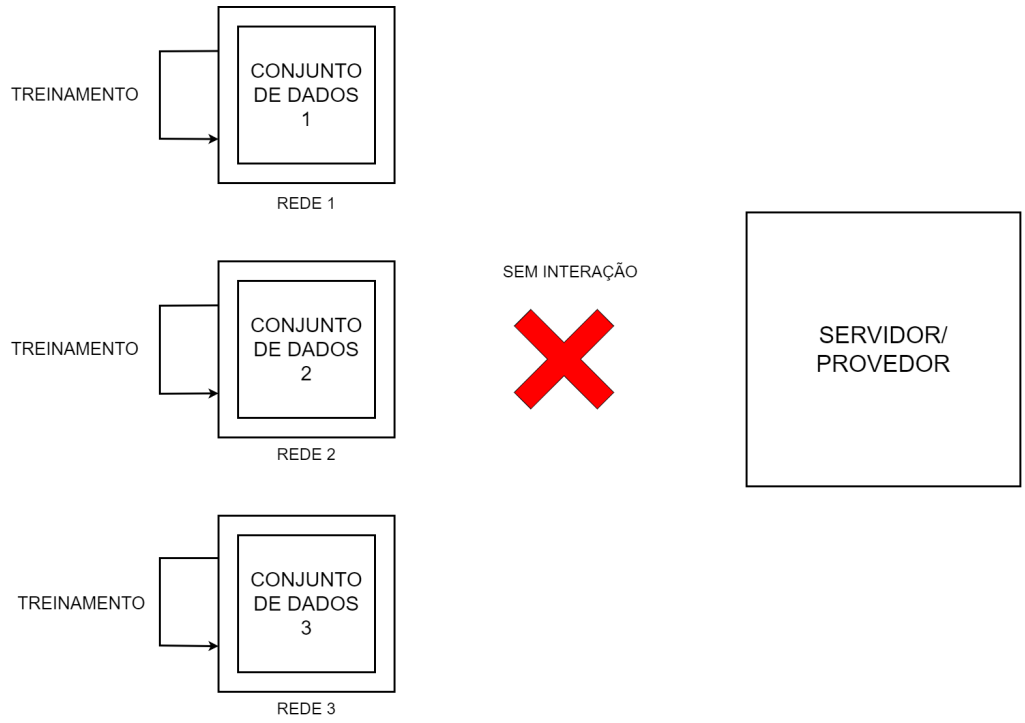

Figura 4 – Diagrama de blocos Aprendizado Local.

O segundo sistema implementado foi o de aprendizado local, ilustrado na Figura 4. Neste cenário foram implementados três algoritmos de aprendizado de máquina usando redes neurais, um para cada conjunto de dados. A inferência também é individual, cada um dos três algoritmos realiza seu treinamento e seus testes sobre o seu conjunto de dados designado.

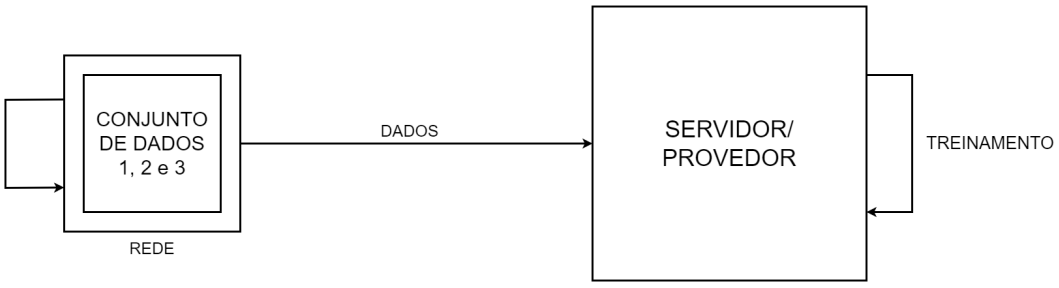

Figura 5 – Diagrama de blocos Aprendizado Centralizado.

Por fim, o terceiro sistema implementado consiste no aprendizado centralizado, ilustrado na figura 5. Como primeiro diferencial, podemos citar o conjunto de dados, neste cenário, os três conjuntos de dados são reunidos em um. O conjunto de dados

resultante utilizado possui todas as amostras dos citados anteriormente organizadas de forma aleatória. Além disso, o treinamento não ocorre na rede local, e sim no servidor. A rede local envia seus dados para um servidor central, que por sua vez realiza o treinamento com os dados recebidos usando uma rede neural. Os testes também são realizados no servidor, com dados recebidos da rede local.

O primeiro cenário foi criado para verificar de que forma a utilização do aprendizado federado pode afetar a eficácia do IDS. Assim, o segundo cenário nos permitiu fazer essa comparação, pois nele temos uma abordagem mais tradicional onde cada rede treina com seus próprios dados e não existe agregação. Por outro lado, o terceiro cenário nos permite fazer uma análise sobre a agregação dos *workers*, o objetivo da agregação é a colaborar para criar um modelo melhor para todos. O último cenário não possui uma etapa de agregação, mas seu modelo recebe diretamente todas as amostras, ao invés de receber três modelos e agregá-los em um.

O modelo de detecção de intrusão proposto é um sistema NIDS, a detecção é realizada por meio da análise do tráfego de rede em busca de padrões de atividades que indiquem a presença de um ataque ou tentativa de invasão. Os dados utilizados serão fluxos de tráfego e seus detalhes podem ser observados na Tabela 1, cada fluxo de tráfego será analisado por uma rede neural que determinará se ele é uma amostra natural ou um ataque.

Tabela 1 – Características extraídas do fluxo.

Nome	**Descrição**
`src_port`	Porta de origem da camada de transporte
`dst_port`	Porta de destino da camada de transporte
`protocol`	Protocolo da camada de transporte
`ip_version`	Versão IP
`tunnel_id`	Identificador de túnel (0: Sem túnel, 1: GTP, 2: CAPWAP, 3: TZSP)
`bidirectional_duration_ms`	Duração do fluxo em milissegundos
`bidirectional_packets`	Acumulador de pacotes do fluxo
`bidirectional_bytes`	Acumulador de bytes do fluxo
`src2dst_duration_ms`	Duração do fluxo em milissegundos a partir da origem
`src2dst_packets`	Acumulador de pacotes do fluxo partir da origem
`src2dst_bytes`	Acumulador de bytes do fluxo a partir da origem
`dst2src_duration_ms`	Duração do fluxo em milissegundos a partir do destino
`dst2src_packets`	Acumulador de pacotes do fluxo partir do destino
`dst2src_bytes`	Acumulador de bytes do fluxo partir do destino

3.2 Conjuntos de Dados

Algoritmos de aprendizado de máquina precisam de conjuntos de dados para aprender a fazer previsões ou decisões. Os conjuntos de dados fornecem ao algoritmo exemplos de entradas e suas saídas correspondentes, permitindo que o algoritmo aprenda a relação entre as entradas e saídas e generalize para novas entradas que nunca viu antes. Quanto mais diversificado e representativo for o conjunto de dados, melhor será o desempenho do algoritmo em dados inéditos.

Dentro da literatura não se encontrou um conjunto de dados focado em aprendizado federado para detecção de intrusões, essa limitação levou à escolha de três conjuntos de dados para a realização deste trabalho. Visando se adaptar da melhor forma possível à ideia do aprendizado federado, o objetivo é que cada um dos conjuntos de dados emule uma rede doméstica. São eles: TON_IOT [2], CIC IoT dataset 2022 [49] e IoT-23 [4]. Além disso, o servidor central faz o papel de um provedor de serviço, recebendo informações sobre o tráfego dessas redes (*workers*) e sendo capaz de dizer se eles estão sendo atacados ou não.

Os conjuntos de dados escolhidos estão disponíveis no formato de captura de rede, facilitando a manipulação do formato dos dados. Com o objetivo de padronizar as $features$ de três conjuntos de dados diferentes, foi criado um programa utilizando a API NFStream. O objetivo do programa é extrair uma série de características pré-determinadas dos arquivos de captura de rede. As *features* escolhidas podes ser vistas na Tabela 1. A saída do programa é um arquivo no formato csv contendo as informações do arquivo de captura no formato desejado.

Após a seleção de características, os arquivos de captura de rede (pcaps) dos três conjuntos de dados selecionados são processados no programa citado anteriormente. Assim, obtendo todos as *features* selecionadas dos fluxos originais em um formato csv, resultando em três conjuntos de dados padronizados. O próximo passo é fazer a rotulação dos conjuntos de dados. Cada conjunto de dados tem suas próprias regras de rotulação, elas serão apresentadas a seguir juntamente com detalhes sobre os conjuntos de dados.

3.2.1 TON_IOT

Os conjuntos de dados TON_IoT podem ser usado para avaliar a fidelidade e a eficiência de vários aplicativos de cibersegurança habilitados para IA. Os conjuntos de dados foram chamados de “TON_IoT”, pois incluem fontes de dados heterogêneas coletadas de sensores IoT e IIoT, sistemas operacionais Windows 7 e 10, como Ubuntu 14.04 e 18.04 LTS e tráfego de rede. Os conjuntos de dados foram coletados de uma rede de teste realista e em larga escala, que foi projetada no Laboratório de IoT da University of New South Wales Canberra Cyber [2].

Gerar cenários normais/benignos e de ataque é a principal tarefa de criar conjuntos de dados autênticos e realistas [50]. Isso aconteceu por meio de operações legítimas dos servidores enquanto eles não iniciavam nenhum evento de ataque. Os sistemas ofensivos incluem as maquinas virtuais Kali Linux e scripts para invadir os sistemas do ambiente de teste da rede. Dez endereços IP estáticos (192.168.1.30-39) foram empregados para lançar os ataques e violar os sistemas vulneráveis ou serviços IoT/IIoT.

A regra de rotulação do conjunto de dados foi feita com base nos IPs de origem. Os criadores definiram que todos os ataques possuiriam como origem um IP entre 192.168.1.30 e 192.168.1.39. Na tabela 2 é possível observar todos os tipos de ataques lançados, e os IPs que lançaram os ataques. Para rotular o conjunto de dados foi necessário analisar todos os fluxos e rotular como "ataque" todos que possuíam como origem um IP entre 192.168.1.30 e 192.168.1.39, os outros fluxos foram rotulados como amostras naturais.

Tabela 2 – Relação entre as nove categorias de ataques e os IPs estáticos do *Kali Linux* que estão propagando os ataques. Fonte: [2]

Attack	**IPs**
`Scanning attack`	192.168.1.{30,31,32,33,38}
`Denial of Service (DoS) attack`	192.168.1.{30,31,39}
`Distributed Denial of Service (DDoS) attack`	192.168.1.{30,31,34,35,36,37,38}
`Ransomware attack`	192.168.1.{33,37}
`Cross-site Scripting (XSS) attack`	192.168.1.{32,35,36,39}
`Password cracking attack`	192.168.1.{30,31,33,35,38}
`Man-In-The-Middle (MITM) attack`	192.168.1.{31,34}
`Backdoor attack`	192.168.1.{33,37}

Após o pré-processamento, o conjunto de dados resultante contava com 27.112.063 fluxos, sendo 25.539.900 ataques e 1.572.163 naturais. Por conta de uma limitação de hardware, foram selecionadas 1.000.000 de amostras aleatórias para compor o conjunto de dados final, mantendo a proporção anterior entre amostras de ataque e naturais.

3.2.2 CIC IoT dataset 2022

O conjunto de dados CIC IoT dataset 2022 teve sua instalação e configuração em um laboratório de experimentos. Foram utilizados 60 dispositivos IoT e diferentes tipos de experimentos. Esses experimentos foram conduzidos no Canadian Institute for Cybersecurity por três meses [49]. Os dispositivos utilizados no experimento podem ser vistos na Tabela 3.

Foram coletados dados de todos os dispositivos conectados à rede. Dependendo do experimento realizado, o tráfego foi capturado para toda a rede, parte da rede, um único dispositivo, ou ainda, filtrado por um ID distinto, como um endereço MAC. Seis experimentos diferentes foram conduzidos nos dispositivos:

1. Power: a captura dura dois minutos, sendo sincronizada com o tempo em que o dispositivo é conectado à fonte de alimentação. Terminado o tempo, o dispositivo é desconectado da fonte de alimentação e a captura continua por mais três minutos para coletar os pacotes restantes;

2. Idle: foram coletados dados de um período onde todo o laboratório foi completamente evacuado e não houve interações humanas envolvidas. Para coletar esses dados em massa, foi utilizado um arquivo em lote. O arquivo em lote contém o script para coletar o tráfego de rede de cada dispositivo usando o *dumpcap*, com cada instância sendo filtrada por seus respectivos endereços MAC;

3. Interactions: neste experimento, todas as funcionalidades possíveis dos dispositivos IoT foram extraídas, atividade de rede correspondente e os pacotes transmitidos para cada funcionalidade/atividade foram capturados. Algumas formas de interações utilizadas foram físicas (como botões), LAN App e WAN App (disponibilizados pelos aparelhos IoT) e voz (Amazon Alexa);

4. Scenarios: nesses experimentos, foram conduzidos experimentos usando uma combinação de dispositivos, como simulações da atividade de rede, dentro de uma casa inteligente. Esses experimentos foram feitos para ver como os dispositivos se comportam enquanto interagem uns com os outros simultaneamente;

5. Active: esse experimento foi realizado durante 30 dias. Durante esse período foi feita a captura de um ambiente considerado ativo, onde pessoas entravam e saiam do laboratório esporadicamente, podendo assim interagir com os dispositivos de forma ativa ou passiva;

6. Attacks: foram realizados dois ataques diferentes em alguns dos dispositivos, ataques Flood e ataques RTSP Brute Force. O ataque Flood foi feito usando uma ferramenta chamada Low Orbit Ion Cannon (LOIC) para interromper o funcionamento normal dos dispositivos, usando diferentes tipos de protocolos de rede, como HTTP, UDP e TCP lançando ataques DoS. Já o ataque RTSP Brute Force foi feito usando outras ferramentas, chamadas Hydra e Nmap, para encontrar e tentar acessar as câmeras conectadas aos dispositivos, usando um método de força bruta.

Como ocorreu com o TON-IoT, também foi necessário rotular os fluxos obtidos deste conjunto de dados. Para tanto os fluxos foram rotulados com base na divisão dos experimentos. Assim, todos os dados na categoria de experimentos “Attack” foram rotulados como ataques. Após o pré-processamento, o conjunto de dados resultante contava com 1.920.125 fluxos, sendo 270.626 ataques e 1.649.499 naturais. Por conta de uma limitação de hardware, foram selecionadas 1.000.000 de amostras aleatórias para compor o conjunto de dados final, mantendo a proporção anterior entre amostras de ataque e naturais.

Tabela 3 – Dispositivos IoT usados no conjunto de dado CIC-IOT. Fonte: [3]

Categoria	Nome do dispositivo	Conexão	Marca
Câmera	HeimVision Smart WiFi Camera	Wi-Fi	HeimVision
	Netatmo Camera	Wi-Fi	Netatmo
	Arlo Q Camera	Wi-Fi	Arlo
	DCS8000LHA1 D-Link Mini Camera	Wi-Fi	D-Link
	Borun/Sichuan-AI Camera	Wi-Fi	BORUN
	AMCREST WiFi Camera	Wi-Fi	AMCREST
	Shenzhen 73:f3:36 Home Eye Camera	Wi-Fi	Shenzhen
	Luohe Cam Dog	Wi-Fi	LUOHE
	SIMCAM 1S (AMPAKTec)	Wi-Fi	SIMCAM
	Nest Indoor Camera	Wi-Fi	Nest
Lâmpadas inteligentes	SmartThings Smart Bulb (5)	Zigbee	SmartThings
	Philips Hue White (2)	Zigbee	Philips
	HeimVision SmartLife Radio/Lamp	Wi-Fi	HeimVision
	Globe Lamp ESP B1680C	Wi-Fi	The Globe Electric
Alto falante	Amazon Echo Studio	Wi-Fi	Amazon
	Amazon Echo Dot (2)	Wi-Fi	Amazon
	Google Nest Mini	Wi-Fi	Google
	Amazon Echo Spot	Wi-Fi	Amazon
	Sonos One Speaker	Wi-Fi	Sonos
Sensor de porta/janela	Eufy HomeBase II	Wi-Fi	Eufy
	Fibaro Door/Window Sensor (3)	Z-Wave	Fibaro
Estação base	Ring Base Station	Wi-Fi	Ring
	Arlo Base Station	por fio	Arlo
Sensor de movimento	AeoTec Motion Sensor	Zigbee	AeoTec
	Fibaro Motion Sensor (5)	Z-Wave	Fibaro
Atuador	SmartThings Button	Zigbee	SmartThings
	AeoTec Button	Zigbee	AeoTec
Multi Sensor	AeoTec Multipurpose Sensor	Zigbee	AeoTec
Estação meteorológica	Netatmo Weather Station	Wi-Fi	Netatmo
	AeoTec Water Leak Sensor	Zigbee	AeoTec
	Fibaro Flood Sensor (2)	Z-Wave	Fibaro
Aspirador	iRobot Roomba	Wi-Fi	iRobot
Smart Hub	Vera Plus Hub	Wi-Fi	Vera Control
	Philips Hue Bridge (2)	Zigbee	Philips
	SmartThings Hub Plug	Zigbee	SmartThings
	Fibaro Home Center Lite	Z-Wave	Fibaro
Tomada inteligente	Yutron Plug (2)	Wi-Fi	Yutron
	Amazon Plug	Wi-Fi	Amazon
	Fibaro Wall Plug (2)	Z-Wave	Fibaro
	Teckin Plug (2)	Wi-Fi	Teckin
Cafeteira	Atomi Coffee Maker	Wi-Fi	Atomi
Smart TV	LGInnote 4e:00:82	Wi-Fi	LGInnote
Lousa Digital	SMARTTec f6:e3:cb	Wi-Fi	SMART Tec

3.2.3 IoT-23

IoT-23 [4] são conjuntos de dados de tráfego de rede de dispositivos da IoT que foram criados pelo laboratório Avast AIC com financiamento da Avast Software. O objetivo principal da criação foi oferecer grandes conjuntos de dados de tráfegos de rede reais de dispositivos IoT, permitindo que pesquisadores desenvolvam algoritmos de aprendizado de máquina para detecção de *malwares*.

O conjunto de dados possui 20 capturas de tráfego em dispositivos infectados por *malware* e três capturas para tráfego benigno de dispositivos IoT. Ele foi publicado pela primeira vez em janeiro de 2020, com capturas variando de 2018 a 2019. Este tráfego de rede IoT foi capturado no Stratosphere Laboratory, Artificial Intelligence Centre group, Faculty of Electrical Engineering, Czech Technical University na República Tcheca.

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

O tráfego de rede capturado para os cenários benignos foi obtido capturando o tráfego de rede de três dispositivos IoT diferentes: uma lâmpada LED inteligente Philips HUE, um assistente pessoal inteligente doméstico Amazon Echo e uma fechadura inteligente Somfy.

Diferente das rotulações vistas nos conjuntos de dados anteriores, o IOT-23 possui uma regra diferente para cada um de seus cenários maliciosos disponibilizadas pela organização criadora do conjunto de dados. A coluna “Field” diz qual das *features* vai ser a base da rotulação, a partir disso, olhando a coluna “Data” vemos o valor que essa *feature* deve ser comparado. A coluna “Comparator” diz como essa comparação deve ser feita, em “Label” vemos o tipo do ataque e “Type” que é Malicioso. A coluna “bro field number” traz um ID para as *features* na coluna “Field”. Por exemplo, na Tabela 4, temos as regra de rotulação do cenário CTU-IoT-Malware-Capture-1-1, ao observar as regras de rotulação impostas, vemos que os fluxos marcados como “ataques”, neste casa, são aqueles que possuem o IP da porta de destino igual a 2407, 9527, 8080, 23 ou 2323.

Tabela 4 – Regras de rotulação para IoT-23. Fonte: [4]

ID	Field	bro field number	Data	Comparator	Label	Type	Connector
1	id.resp_p	6	2407	“eq”	C & C	Malicious	-
2	id.resp_p	6	9527	“eq”	Part Of A Horizontal Port Scan	Malicious	-
3	id.resp_p	6	8080	“eq”	Part Of A Horizontal Port Scan	Malicious	-
4	id.resp_p	6	23	“eq”	Part Of A Horizontal Port Scan	Malicious	-
5	id.resp_p	6	2323	“eq”	Part Of A Horizontal Port Scan	Malicious	-

Após o pré-processamento, o conjunto de dados resultante contava com 35.891.706 fluxos, sendo 29.356.100 ataques e 6.535.606 naturais. Por conta de uma limitação de hardware, foram selecionadas 1.000.000 de amostras aleatórias para compor o conjunto de dados final, mantendo a proporção anterior entre amostras de ataque e naturais.

3.3 Redes Neurais

Neste trabalho foram implementados três sistemas e todos usam redes neurais. O primeiro sistema usa o aprendizado federado, o segundo sistema usa um aprendizado local, e o terceiro sistema utiliza o aprendizado centralizado. Foram realizados vários experimentos para os três sistemas utilizando diferentes topologias de redes neurais para verificar qual topologia poderia proporcionar o melhor desempenho preditivo.

Por consequência do número de *features* do conjunto de dados, a camada de entrada sempre tem 14 neurônios e, por se tratar de um problema de classificação binária, a camada de saída sempre tem 1 neurônio. Foram analisadas topologias com uma camada oculta, com quantidade de neurônios variando entre 6 e 30 neurônios. Ainda, foram testadas topologias com duas camadas ocultas, sendo que a primeira camada possuía 30 neurônios e a segunda variando entre 15 e 60. Por fim, foi analisada uma topologia com três camadas ocultas, sendo que a primeira e terceira camada contêm 30 neurônios e a segunda camada 60.

Outros componentes importantes em uma rede neural são as funções de ativação, neste trabalho, optou-se por trabalhar com a função de ativação ReLU (Rectified Linear Unit - Ativação Linear Retificada) nas camadas ocultas. A função de ativação ReLU está

presente após cada uma das camadas da rede neural, com exceção da última camada, pois optou-se por utilizar uma função de ativação Sigmoid na camada de saída. Além das funções de ativação, outros parâmetros importantes são a função de perda e o otimizador. A função de perda selecionada foi a *Binary Cross Entropy*, e em relação ao otimizador, a escolha foi o *Stochastic Gradient Descent*, esse conjunto de funções foi utilizado para todas as topologias testadas.

3.4 Implementação

Como discutido na seção 3.1, foram implementados três modelos de detecção de intrusão. A implementação desses sistemas foi feita utilizando a linguagem Python e as bibliotecas Pytorch [51] e PySyft [52].

A divisão entre dados de treinamento e teste foi de 80% e 20%. Todos os algoritmos começam com um modelo de rede neural tendo seus pesos e vieses inicializados de forma aleatória. A partir disso eles passam por um ciclo iterativo de alimentação de exemplos de treinamento para a rede, cálculo das previsões da rede e ajuste dos pesos e vieses da rede para minimizar o erro de predição.

Para o treinamento e para o teste, o tamanho escolhido para *batch* foi 64. Outro parâmetro utilizado foi o *learning rate* igual a 0,01. O erro de predição é medido pela função de perda (em todos os casos, a *Binary Cross Entropy*), que calcula a diferença entre as previsões da rede e as saídas desejadas para os exemplos de treinamento.

Durante o treinamento, o algoritmo de aprendizado usa o gradiente da função de perda para atualizar os valores dos pesos e vieses por meio de um processo conhecido como descida de gradiente. Os treinamentos foram realizados por 100 épocas, sendo que a cada 20 épocas temos uma etapa de teste. No caso do aprendizado federado, a agregação também é realizada a cada 20 épocas, logo antes da etapa de teste.

4 RESULTADOS

Este capitulo apresenta os resultados para as métricas de desempenho coletadas a partir dos experimentos realizados com os três modelos de detecção de intrusão detalhados em na seção 3.1. Na seção 4.1 estão os resultados do aprendizado federado, sistema que busca o aprendizado de maneira colaborativa e descentralizada. Na seção 4.2 podem ser vistos os resultados do aprendizado local, que implementa três algoritmos, um para cada conjunto de dados. A seção 4.3 traz os resultados de aprendizado centralizado, um paradigma mais tradicional dentro do aprendizado de máquina. Por fim, na seção 4.4 os resultados apresentados anteriormente são analisados e discutidos.

4.1 Aprendizado Federado

Os resultados obtidos pelo aprendizado federado podem ser observados na Tabela 5, ela traz três métricas: F1-score, *precision* e *recall*. Podemos visualizar os resultados para cada um dos *workers* em cada uma das topologias de redes neurais testadas. A coluna de "Camadas ocultas" mostra o número de neurônios que cada camada oculta possui. Com o treinamento completo, o sistema teve seus resultados coletados individualmente para cada *worker*.

Tabela 5 – Resultados do sistema que utilizou aprendizado federado.

Camadas ocultas	**Métrica**	**CIC-IOT**	**TON-IOT**	**IOT-23**
[6]	F1-score	0,0434	0,4995	0,8962
	Precision	0,0300	0,7727	0,8127
	Recall	0,0775	0,3690	0,9987
[15]	F1-score	0,4145	0,5350	0,9483
	Precision	0,8040	0,5968	0,9102
	Recall	0,2792	0,4847	0,9896
[30]	F1-score	0,4681	0,4459	0,8713
	Precision	0,6713	0,4115	0,7810
	Recall	0,7020	0,2299	0,8193
[30, 15]	F1-score	0,1602	0.6314	0.8144
	Precision	0,1062	0.6083	0.7870
	Recall	0,3259	0.6563	0.8437
[30, 3]	F1-score	0,6864	0,2950	0,7997
	Precision	0,6713	0,4115	0,7810
	Recall	0,7020	0,2299	0,8193
[30, 60]	F1-score	0,5547	0,6360	0,7112
	Precision	0,3999	0,7853	0,7475
	Recall	0,9048	0,5343	0,6782
[30, 60, 30]	F1-score	0,6836	0,4213	0,6000
	Precision	0,5485	0,5234	0,6969
	Recall	0,9068	0,3525	0,5267

Analisando os resultados da F1-score, vemos que inicialmente os resultados para o CIC-IOT são baixíssimos (0.0434), enquanto os do TON-IOT são ruins, estando em 0,4995. Já o IOT-23 produz resultados melhores, em torno de 0,8962. Conforme a rede neural fica mais complexa, os resultados da F1-score se alteram significativamente, em alguns casos temos uma melhora tanto do CIC-IOT quanto do TON-IOT, indo para aproximadamente 0,5, ao mesmo tempo que o IOT-23 mantém um resultado de 0,9483.

A partir deste ponto, quanto mais complexa a rede neural fica, mais os resultados do IOT-23 caem, chegando a 0,7. Enquanto isso, os outros *workers* não seguem uma regra, o TON-IOT, que até então se mantinha perto de 0,5 atinge o seu pior resultado de 0,2950, e o CIC-IOT mantém um resultado próximo a 0,6. Existe apenas um cenário em que os três *workers* tem um pontuação acima de 0,5 simultaneamente, isso ocorre quando a rede neural tem duas camadas ocultas, uma com 30 neurônios e outra com 60 neurônios. Neste caso, o CIC-IOT atinge 0,5547, o TON-IOT atinge 0,6360 e o IOT-23 atinge 0,7112. Considerando a F1-score, o melhor resultado obtido pelo CIC-IOT é de

0,6864, pelo TON-IOT é 0,6360 e pelo IOT-23 é 0,9483.

Em relação às métricas *precision* e *recall*, inicialmente o CIC-IOT apresenta resultados baixíssimos para ambos. Já o TON-IOT tem uma alta *precision* mas uma *recall* baixa, mostrando que ele tem um baixo número de falsos positivos, mas deixa de classificar corretamente muitas amostras maliciosas. O IOT-23 mantém suas métricas bem altas, sua *recall* é de 0,9987 e sua *precision* é de 0,8127.

Assim como nos resultados da F1-score, é difícil achar um momento onde os três *workers* têm bons resultados em ambas *precision* e *recall*. No fim dos testes, o CIC-IOT chega a ter uma *recall* acima de 0,9, mas a sua *precision* é muito baixa (0,5485), indicando que o limiar de classificação entre naturais e ataques está pendendo para o lado de ataques. Enquanto isso, o TON-IOT têm resultados medianos para as duas métricas, em variando entre 0,6 e 0,4. O IOT-23 também fica com resultados medianos, com uma *precision* em torno de 0,6969 e uma *recall* 0,5267, indicando que ele está deixando de classificar amostras que eram positivas e chegando no seu pior desempenho até então.

4.2 Aprendizado Local

Os resultados obtidos pelo aprendizado local podem ser observados na Tabela 6, ela traz três métricas: F1-score, *precision* e *recall*. Podemos visualizar os resultados do treinamento de cada um dos conjuntos de dados em cada uma das topologias de redes neurais testadas. A coluna de “Camadas ocultas” mostra o número de neurônios que cada camada oculta possui. Neste cenário, são apresentados os resultados de três algoritmos, pois cada conjunto de dados recebeu sua própria implementação de aprendizado local.

Tabela 6 – Resultados do sistema que utilizou aprendizado local.

Camadas ocultas	**Métrica**	**CIC-IOT**	**TON-IOT**	**IOT-23**
[6]	F1-score	0,8941	0,8910	0,7808
	Precision	0,8316	0,8142	0,7752
	Recall	0,9668	0,9838	0,7865
[15]	F1-score	0,7399	0,8886	0,8007
	Precision	0,9055	0,8102	0,7822
	Recall	0,6254	0,9837	0,8199
[30]	F1-score	0,8988	0,9060	0,7912
	Precision	0,9222	0,8742	0,7788
	Recall	0,8764	0,9400	0,8039
[30, 15]	F1-score	0,9221	0,9098	0,7818
	Precision	0,9487	0,8934	0,7792
	Recall	0,8969	0,9268	0,8046
[30, 30]	F1-score	0,9175	0,9104	0,7686
	Precision	0,9183	0,8909	0,7706
	Recall	0,9167	0,9307	0,7664

Os resultados obtidos para o CIC-IoT demonstram que ele teve um desempenho preditivo satisfatório. Logo no início podemos observar uma F1-score de 0,8941. Conforme a rede neural fica mais complexa, seus resultados melhoram, tendo seu ápice com uma pontuação igual a 0,9221 quando a rede neural possui duas camadas ocultas, uma com 30 neurônios e outra com 15 neurônios. Após esse ponto, é possível perceber que os resultados não melhoram de maneira significativa e ficam próximos a 0,9. Em relação a *precision* e *recall*, não vemos uma diferença muito grande entre as duas métricas, seus valores são bons e parecidos entre si.

Observando os resultados do TON-IOT, eles seguem uma linha parecida com os do CIC-IOT. Mesmo com a rede neural mais simples testada, ele já apresenta uma F1-score de 0,8910. Não é possível ver um ponto onde sua F1-score diminui, quanto mais complexa a rede neural fica, mais altos são os resultados que sua F1-score atinge. Sendo assim, o TON-IOT atinge seu melhor resultado na última topologia de rede neural testada. Considerando as métricas *precision* e *recall*, também não existe uma amplitude muito grande entre os dois valores, seus números são positivos e parecidos entre si. Assim, os valores de F1-score, *precision* e *recall* também são similares.

Por fim, ao realizar uma análise sobre os resultados do IOT-23, podemos observar que eles são bons, porém não são melhores que os resultados obtidos no aprendizado federado. Inicialmente, ele mostra uma F1-score de 0,7808, e seu melhor resultado de 0,8007 se da quando a topologia da rede neural possui uma camada oculta de 15 neurônios. Assim

como os dois conjuntos de dados apresentados anteriormente, suas métricas *precision* e *recall* têm valores parecidos.

4.3 Aprendizado Centralizado

Os resultados obtidos para o aprendizado centralizado podem ser observados na Tabela 7. A tabela conta com testes realizados em diferentes topologias de redes neurais, a coluna "Camadas ocultas" mostra o número de neurônios das camadas ocultas da topologia em questão. Foram avaliadas três métricas: F1-score, *precision* e *recall*. O aprendizado centralizado tem um diferencial no conjunto de dados. Seu conjunto de dados é uma união do CIC-IOT, TON-IOT e IOT-23, totalizando assim, 3.000.000 de amostras dispostas de forma aleatória.

Tabela 7 – Resultados do sistema que utilizou aprendizado centralizado.

Camadas ocultas	**Métrica**	**Geral**
[6]	F1-score	0,8983
	Precision	0,8647
	Recall	0,9345
[15]	F1-score	0,9164
	Precision	0,9026
	Recall	0,9306
[30]	F1-score	0,9010
	Precision	0,8722
	Recall	0,9317
[30, 15]	F1-score	0,8259
	Precision	0,8571
	Recall	0,7968

Podemos observar que os resultados da F1-score são positivos. Utilizando apenas uma camada oculta de 6 neurônios, ele atinge o valor 0,8983. O melhor resultado obtido é 0,9164, quando a topologia da rede neural possuía uma camada oculta de 15 neurônios. Após esse ponto, seus resultados permanecem em uma média de 0,9. Ao analisarmos as métricas de *precision* e *recall*, percebemos que elas atingem valores altos e próximos entre si. Isso faz com que os valores de F1-score, *precision* e *recall* sejam semelhantes.

4.4 Discussão

No geral, o aprendizado federado funcionou bem para o IOT-23, apresentando bons resultados, melhores do que ele mostrou no aprendizado local, mostrando que a

colaboração com os outros *workers* foi um ponto positivo. Porém, para os outros dois *workers* que representaram o CIC-IOT e o TON-IOT, os resultados não foram satisfatórios, suas métricas durando o aprendizado local foram melhores, mostrando que eles não se beneficiaram tando dessa colaboração.

Considerando o cenário de aprendizado local, em comparação com o sistema de aprendizado federado, tanto CIC-IOT quanto TON-IOT obtiveram resultados melhores. Já o conjunto de dados IOT-23, teve melhores resultados com o aprendizado federado. Agora considerando o cenário centralizado, podemos observar que ele teve suas métricas próximas dos melhores resultados obtidos para todos os conjuntos de dados.

As métricas obtidas para os conjuntos de dados CIC-IOT e TON-IOT mostraram valores similares no aprendizado local quando comparado com o centralizado. No entanto, quando realizamos a mesma comparação para o conjunto de dados IOT-23, notamos uma grande discrepância, uma vez que o IOT-23 teve seus piores resultados durante o aprendizado local. Porém, ao comparam as métricas obtidas pelo IOT-23 no aprendizado federado com o aprendizado centralizado, elas são bem próximas.

Sendo assim, observa-se que o aprendizado federado não funciona bem em todos os casos, apresentando dificuldade em lidar com redes muito diferentes, e não encontrando um ponto de equilíbrio entre os *workers*. Além disso, o aprendizado federado exige maior custo de hardware para a sua execução. Ainda, é importante observar que redes com um grande volume de ataques repetidos podem se beneficiar mais de redes com ataques mais variados. Ademais, a agregação por média não obteve bons resultados nesse caso, visto que dois dos três *workers* tiveram resultados ruins se comparados aos modelos de aprendizado local. Neste contexto, é importante considerar que a colaboração da agregação nem sempre é o suficiente, sendo melhor uma abordagem mais tradicional de aprendizado. Desta forma, é importante escolher um método de agregação e um ambiente adequado.

5 CONCLUSÃO

Como consequência da popularização da Internet, as intrusões em redes comerciais e privadas têm sido motivo de preocupação. Portanto, são necessárias medidas para evitar que agentes mal-intencionados prejudiquem outras pessoas e organizações. Para realizar a detecção dessas intrusões o aprendizado de máquina vem sendo amplamente utilizado, pois ele permite criar sistemas facilmente adaptáveis a diferentes realidades e contextos.

Apesar de seu potencial na detecção de intrusões, uso do aprendizado de máquina pode levantar questões sobre a privacidade das informações usadas em seu treinamento. A privacidade torna-se cada vez mais importante, gerando diversas discussões sobre como preservá-la. Neste trabalho, é estudado o uso do aprendizado federado, um paradigma que ajuda a proteger a privacidade.

Com o objetivo de avaliar o uso do aprendizado federado em IDS e comparar diferentes modelos de detecção de intrusão, foram desenvolvidos três sistemas: um que utiliza aprendizado federado, um com aprendizado local e um com aprendizado centralizado. Os experimentos deste trabalho foram realizados com os conjuntos de dados CIC-IOT, TON-IOT E IOT-23. Estes conjuntos de dados passaram por uma série de pré-processamentos e no fim foram selecionadas 1.000.000 de amostras de cada um. Estas amostras foram organizadas de maneira aleatória, mas mantiveram as proporções originais das classes.

Com os três sistemas implementados, seus resultados foram reunidos e analisados. Os parâmetros avaliados incluem: número de verdadeiros positivos, falsos positivos, verdadeiros negativos e falsos negativos, *precision*, *recall* e F1 score. Os resultados obtidos nos diferentes sistemas de aprendizado variaram conforme os conjuntos de dados analisados. No caso do IOT-23, o aprendizado federado se mostrou eficaz, enquanto para os outros conjuntos de dados o sistema de aprendizado local teve um desempenho superior. Durante o aprendizado local, tanto o CIC-IOT quanto o TON-IOT alcançaram uma F1-score acima de 90%, indicando resultados satisfatórios. Já o sistema de aprendizado centralizado também obteve uma F1-score superior a 90%, deixando uma pequena margem de diferença em relação aos outros sistemas.

Por fim, concluo que neste trabalho foi possível avaliar um modelo de detecção de intrusão que utiliza aprendizado federado e compará-lo com outros dois modelos. Pode-se dizer que ele obteve resultados satisfatórios em detectar ataques em apenas um de seus três *workers*. Enquanto os conjuntos de dados que compunham os outros dois *workers* obtiveram mais sucesso em um modelo de detecção de intrusão local. O aprendizado federado não é a melhor opção em todos os casos, mas quando possível ele traz uma maneira de aprender colaborativamente com segurança e privacidade.

REFERÊNCIAS

[1] SHINGI, G.; SAGLANI, H.; JAIN, P. *Segmented Federated Learning for Adaptive Intrusion Detection System*. arXiv, 2021. Disponível em: <https://arxiv.org/abs/2107.00881>.

[2] MOUSTAFA, N. A new distributed architecture for evaluating ai-based security systems at the edge: Network ton_iot datasets. *Sustainable Cities and Society*, v. 72, p. 102994, 2021. ISSN 2210-6707. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2210670721002808>.

[3] DADKHAH, S. et al. Towards the development of a realistic multidimensional iot profiling dataset. In: *2022 19th Annual International Conference on Privacy, Security & Trust (PST)*. [S.l.: s.n.], 2022. p. 1–11.

[4] GARCIA, S.; PARMISANO, A.; ERQUIAGA, M. J. *IoT-23: A labeled dataset with malicious and benign IoT network traffic*. Zenodo, 2020. More details here https://www.stratosphereips.org /datasets-iot23. Disponível em: <https://doi.org/10.5281/zenodo.4743746>.

[5] TSAI, C.-F. et al. Intrusion detection by machine learning: A review. *Expert Systems with Applications*, v. 36, n. 10, p. 11994–12000, 2009. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417409004801>.

[6] HARTMANN, F. Federated learning. *línea]. Available: https://florian. github. io/federated-learning/.[Último acceso: 15 10 2019]*, 2018.

[7] NIKSEFAT, S.; KAGHAZGARAN, P.; SADEGHIYAN, B. Privacy issues in intrusion detection systems: A taxonomy, survey and future directions. *Computer Science Review*, v. 25, 08 2017.

[8] YANG, Q. et al. Federated learning. *Morgan and Claypool*, 2019.

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

[10] AL-MARRI, N. A. A.-A.; CIFTLER, B. S.; ABDALLAH, M. M. Federated mimic learning for privacy preserving intrusion detection. In: *2020 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*. [S.l.: s.n.], 2020. p. 1–6.

[11] ZHU, H.; JIN, Y. Multi-objective evolutionary federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, v. 31, n. 4, p. 1310–1322, 2020.

[12] BERTOLI, G. de C.; JúNIOR, L. A. P.; SAOTOME, O. Improving detection of scanning attacks on heterogeneous networks with federated learning. Association for Computing Machinery, New York, NY, USA, v. 49, n. 4, p. 118–123, jun 2022. ISSN 0163-5999. Disponível em: <https://doi.org/10.1145/3543146.3543172>.

[13] SUN, Y.; ESAKI, H.; OCHIAI, H. Adaptive intrusion detection in the networking of large-scale lans with segmented federated learning. *IEEE Open Journal of the Communications Society*, v. 2, p. 102–112, 2021.

[14] PREUVENEERS, D. et al. Chained anomaly detection models for federated learning: An intrusion detection case study. *Applied Sciences*, 2018.

[15] ZHAO, Y. et al. Multi-task network anomaly detection using federated learning. In: . New York, NY, USA: Association for Computing Machinery, 2019. (SoICT 2019), p. 273–279. ISBN 9781450372459. Disponível em: <https://doi.org/10.1145/3368926.3369705>.

[16] BRIGGS, C.; FAN, Z.; ANDRAS, P. Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. [S.l.: s.n.], 2020. p. 1–9.

[17] WANG, H. et al. Optimizing federated learning on non-iid data with reinforcement learning. In: *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. [S.l.: s.n.], 2020. p. 1698–1707.

[18] NGUYEN, T. D. et al. DÏot: A federated self-learning anomaly detection system for iot. In: *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. [S.l.: s.n.], 2019. p. 756–767.

[19] REY, V. et al. Federated learning for malware detection in iot devices. *Computer Networks*, v. 204, p. 108693, 2022. ISSN 1389-1286. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1389128621005582>.

[20] ASHTON, K. et al. That 'internet of things' thing. *RFID journal*, Hauppauge, New York, v. 22, n. 7, p. 97–114, 2009.

[21] HALLER, S. The things in the internet of things. *Poster at the (IoT 2010). Tokyo, Japan, November*, v. 5, n. 8, p. 26–30, 2010.

[22] ROSE, K.; ELDRIDGE, S.; CHAPIN, L. The internet of things: An overview. *The internet society (ISOC)*, Reston, VA, v. 80, p. 1–50, 2015.

[23] MUSSIO, L. D.; MAIA, R. F.; LOPES, G. W. Um estudo de arquiteturas iot para dispositivos embarcados.

[24] MELL, P.; GRANCE, T. et al. The nist definition of cloud computing. Computer Security Division, Information Technology Laboratory, National ..., 2011.

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

[26] BONOMI, F. et al. Fog computing and its role in the internet of things. In: *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. [S.l.: s.n.], 2012. p. 13–16.

[27] ZARPELãO, B. B. et al. A survey of intrusion detection in internet of things. *Journal of Network and Computer Applications*, v. 84, p. 25–37, 2017. ISSN 1084-8045. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1084804517300802>.

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

[29] NAKAMURA, E. T.; GEUS, P. L. de. *Segurança de redes em ambientes cooperativos*. [S.l.]: Novatec Editora, 2007.

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

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

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

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

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

[35] MITCHELL, T. et al. Machine learning. *Annual Review of Computer Science*, v. 4, n. 1, p. 417–433, 1990. Disponível em: <https://doi.org/10.1146/annurev.cs.04.060190.002221>.

[36] KOTSIANTIS, S.; ZAHARAKIS, I.; PINTELAS. Machine learning: a review of classification and combining techniques. *Artif Intell Rev*, v. 26, n. 1, p. 159–190, 2006. Disponível em: <https://doi.org/10.1007/s10462-007-9052-3>.

[37] MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. *Foundations of machine learning*. [S.l.: s.n.], 2018.

[38] WANG, S.-C.; WANG, S.-C. Artificial neural network. *Interdisciplinary computing in java programming*, Springer, p. 81–100, 2003.

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

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

[41] LIPPMANN, R. An introduction to computing with neural nets. *IEEE Assp magazine*, IEEE, v. 4, n. 2, p. 4–22, 1987.

[42] FEDERATED Learning: Collaborative Machine Learning without Centralized Training Data. Https://ai.googleblog.com/2017/04/federated-learning-collaborative.html. Accessed: 2022-13-08.

[43] 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.

[44] LI, T. et al. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, v. 2, p. 429–450, 2020.

[45] YU, P.; WYNTER, L.; LIM, S. H. Fed+: A family of fusion algorithms for federated learning. *CoRR*, abs/2009.06303, 2020. Disponível em: <https://arxiv.org/abs/2009.06303>.

[46] REDDI, S. J. et al. Adaptive federated optimization. *CoRR*, abs/2003.00295, 2020. Disponível em: <https://arxiv.org/abs/2003.00295>.

[47] LIU, Y.; DACHMAN-SOLED, D.; SRIVASTAVA, A. Mitigating reverse engineering attacks on deep neural networks. In: *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. [S.l.: s.n.], 2019. p. 657–662.

[48] LI, B. et al. Deepfed: Federated deep learning for intrusion detection in industrial cyber–physical systems. *IEEE Transactions on Industrial Informatics*, v. 17, n. 8, p. 5615–5624, 2021.

[49] DADKHAH, S. et al. Towards the development of a realistic multidimensional iot profiling dataset. In: *2022 19th Annual International Conference on Privacy, Security & Trust (PST)*. [S.l.: s.n.], 2022. p. 1–11.

[50] MOUSTAFA, N.; TURNBULL, B.; CHOO, K.-K. R. Towards automation of vulnerability and exploitation identification in iiot networks. In: IEEE. *2018 IEEE International Conference on Industrial Internet (ICII)*. [S.l.], 2018. p. 139–145.

[51] PASZKE, A. et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, v. 32, 2019.

[52] ZILLER, A. et al. Pysyft: A library for easy federated learning. *Federated Learning Systems: Towards Next-Generation AI*, Springer, p. 111–139, 2021.