Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

A aplicabilidade do aprendizado federado no treinamento de modelos de IA em dados descentralizados

Autor: Victor Amaral Cerqueira
Orientador: Prof. Dr. Fabiano Araujo Soares
Co-Orientador: Eng. Daniel Araújo Miranda, Mestre

Brasília, DF
2023

Victor Amaral Cerqueira

A aplicabilidade do aprendizado federado no treinamento de modelos de IA em dados descentralizados

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Dr. Fabiano Araujo Soares

Coorientador: Eng. Daniel Araújo Miranda, Mestre

Brasília, DF

2023

Victor Amaral Cerqueira
A aplicabilidade do aprendizado federado no treinamento de modelos de IA em dados descentralizados/ Victor Amaral Cerqueira. – Brasília, DF, 2023-
80 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Fabiano Araujo Soares

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2023.

1. Inteligencia Artificial. 2. Privacidade de dados. I. Prof. Dr. Fabiano Araujo Soares. II.Eng. Daniel Araújo Miranda, Mestre. III. Universidade de Brasília. IV. Faculdade UnB Gama. V. A aplicabilidade do aprendizado federado no treinamento de modelos de IA em dados descentralizados

CDU 02:141:005.6

Victor Amaral Cerqueira

A aplicabilidade do aprendizado federado no treinamento de modelos de IA em dados descentralizados

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, :

Prof. Dr. Fabiano Araujo Soares
Orientador

Eng. Daniel Araújo Miranda, Mestre
Co-Orientador

Prof. Cristiano Jacques Miosso, PhD
Convidado 1

Eng. Rafael Santos Ortis, Mestre
Convidado 2

Brasília, DF
2023

Agradecimentos

Meus agradecimentos vão primeiramente e principalmente a minha família, que sempre me apoiou durante minha trajetória e que apesar dos problemas sempre estão presentes e providenciaram uma educação de qualidade além de uma boa criação.

Também gostaria de agradecer a todos os Amigos, conhecidos e colegas, principalmente aos mais próximos que estão comigo desde minha infância e que me ajudaram de diferentes formas ao longo do processo de graduação na Universidade de Brasília, incluindo durante a escrita desta monografia.

Gostaria também de reservar uma parte desta seção para agradecer a Isabella, mulher da minha vida, pois sem ela com toda certeza deste mundo eu não seria capaz de iniciar e menos ainda concluir este trabalho. Obrigado por toda a sua paciência e carinho que teve comigo durante essa fase que você mais do que ninguém sabe o quão desafiador foi.

Por fim, gostaria de agradecer ao orientador Prof. Dr. Fabiano Araujo Soares e ao Co-orientador Daniel Araujo Miranda por todo o apoio durante a escrita do trabalho assim como ensinamentos muito valiosos durante a produção deste trabalho.

Resumo

O crescente foco na preservação da privacidade dos dados na era digital tem impulsionado a adoção de abordagens inovadoras no treinamento de modelos de inteligência artificial (IA), especialmente em cenários onde os dados são descentralizados. O aprendizado federado (AF) emergiu como uma solução promissora para treinar modelos de IA em ambientes descentralizados, mantendo a privacidade dos dados dos usuários. Este trabalho tem como objetivo explorar o potencial do aprendizado federado para superar os desafios associados ao treinamento de modelos de IA em dados distribuídos e garantir a proteção da privacidade. A metodologia adotada envolveu uma abordagem prática e comparativa. Inicialmente, foram conceituadas as estratégias de aprendizado federado e as técnicas de preservação da privacidade, além das técnicas de segurança necessárias para proteger dados sensíveis. O estudo também abordou os principais desafios do uso de dados descentralizados. Em seguida, foram implementados modelos de aprendizado centralizado utilizando a base de dados Fashion-MNIST com diferentes técnicas de treinamento como Redes Neurais Convolucionais (CNN), Perceptrons Multicamadas (MLP) rasos e Perceptrons Multicamadas profundas que depois foram comparados com um modelo de aprendizado federado utilizando a técnica de rede neural convolucional para efetuar o treinamento de cada dispositivo a ser agregado no modelo global. O treinamento e a avaliação de desempenho foram conduzidos utilizando TensorFlow Federated e Google Colab, com foco na análise de métricas como acurácia, perda e acurácia top-3. Os resultados obtidos mostraram que os modelos centralizados, especialmente a CNN, alcançaram um desempenho superior em termos de acurácia e eficiência computacional em relação ao modelo federado. No entanto, o modelo federado, embora tenha enfrentado desafios com tempos de treinamento mais longos e maior complexidade na comunicação entre dispositivos, apresentou resultados promissores na preservação da privacidade, destacando-se em cenários onde a descentralização dos dados é fundamental. As dificuldades ao desenvolver o modelo federado evidenciaram que, apesar de um desempenho competitivo, o aprendizado federado requer otimizações adicionais para melhorar a eficiência computacional. Em conclusão, o trabalho destaca a viabilidade do aprendizado federado como uma abordagem para manter a privacidade em cenários descentralizados, mas também ressalta a necessidade de mais pesquisas para superar suas limitações e otimizar sua aplicação prática.

Palavras-chave: Aprendizado federado, privacidade dos dados, aprendizado de máquina, CNN, descentralização de dados, TensorFlow Federated.

Abstract

The growing focus on data privacy in the digital age has driven the adoption of innovative approaches to training artificial intelligence (AI) models, particularly in scenarios where data is decentralized. Federated learning (FL) has emerged as a promising solution for training AI models in decentralized environments while preserving user data privacy. This work aims to explore the potential of federated learning to overcome the challenges associated with training AI models on distributed data and ensuring privacy protection. The adopted methodology involved a practical and comparative approach. Initially, strategies of federated learning and privacy-preserving techniques were conceptualized, as well as the security techniques necessary to protect sensitive data. The study also addressed the main challenges of using decentralized data. Next, centralized learning models were implemented using the Fashion-MNIST dataset with different training techniques such as Convolutional Neural Networks (CNN), shallow Multilayer Perceptrons (MLP), and deep Multilayer Perceptrons, which were then compared to a federated learning model using a convolutional neural network technique to train each device that would be aggregated into the global model. Training and performance evaluation were conducted using TensorFlow Federated and Google Colab, focusing on metrics such as accuracy, loss, and top-3 accuracy. The results showed that centralized models, particularly the CNN, achieved superior performance in terms of accuracy and computational efficiency compared to the federated model. However, the federated model, while facing challenges with longer training times and increased communication complexity between devices, showed promising results in privacy preservation, standing out in scenarios where data decentralization is essential. The difficulties encountered in developing the federated model highlighted that, despite competitive performance, federated learning requires additional optimizations to improve computational efficiency. In conclusion, the study highlights the feasibility of federated learning as an approach to maintaining privacy in decentralized scenarios, but also emphasizes the need for further research to overcome its limitations and optimize its practical application.

Keywords: Federated learning, data privacy, machine learning, CNN, data decentralization, TensorFlow Federated.

Lista de ilustrações

Lista de abreviaturas e siglas

IA *Inteligência artificial*

ML *Machine Learning*

AM *Aprendizado de Máquina*

FL *Federated Learning*

AF *Aprendizado Federado*

RNA *Redes Neural Artificial*

IOT *Internet of things*

Sumário

1 Introdução

Neste capítulo, é apresentada uma breve Contextualização, no intuito de deixar mais claro sobre o domínio no qual o trabalho está inserido, bem como para apresentar o problema a ser tratado no mesmo. Neste sentido, é acordada uma definição para privacidade de dados, uma vez que se pretende contribuir para um acompanhamento mais adequado desse topico durante o desenvolvimento do sistema.

Sendo assim, os domínios de interesse são Aprendizado federado e Privacidade de dados. Pretende-se desenvolver um modelo de treinamento de aprendizado federado e outro modelo com treinamento centralizado a fim de comparar métricas obtidas ao final de cada treinamento. Na sequência, têm-se a Justificativa para a realização do trabalho e os Objetivos a serem atingidos. Por fim, há uma breve noção quanto aos materiais e métodos adotadas no trabalho, e a apresentação da Organização da Monografia.

1.1 Contextualização

A Inteligência Artificial (IA) tem se tornado uma presença cada vez mais marcante em nossas vidas, permeando desde assistentes virtuais até sistemas de recomendação personalizados e sua utilização na indústria, como pode ser visto nos índices de investimentos corporativos globais relacionados a IA (Figura 1).

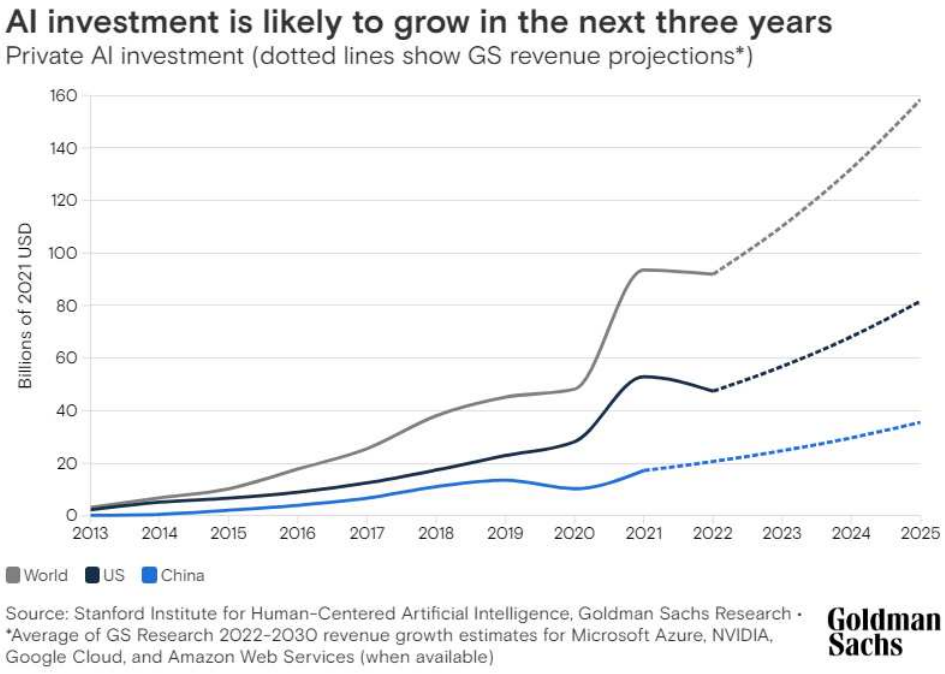


Figura 1 – Investimento em IA no setor privado.

Essa crescente utilização desses modelos reflete a busca incessante por soluções que otimizem processos, tomem decisões e aprimorem a experiência do usuário, como é refletido nas expectativas sobre o impacto da adoção de modelos de inteligência artificial no mercado de trabalho para os próximos anos, Figura 2. No entanto, essa onipresença da IA também traz consigo desafios, especialmente quando se trata do tratamento de dados sensíveis e da privacidade dos usuários.

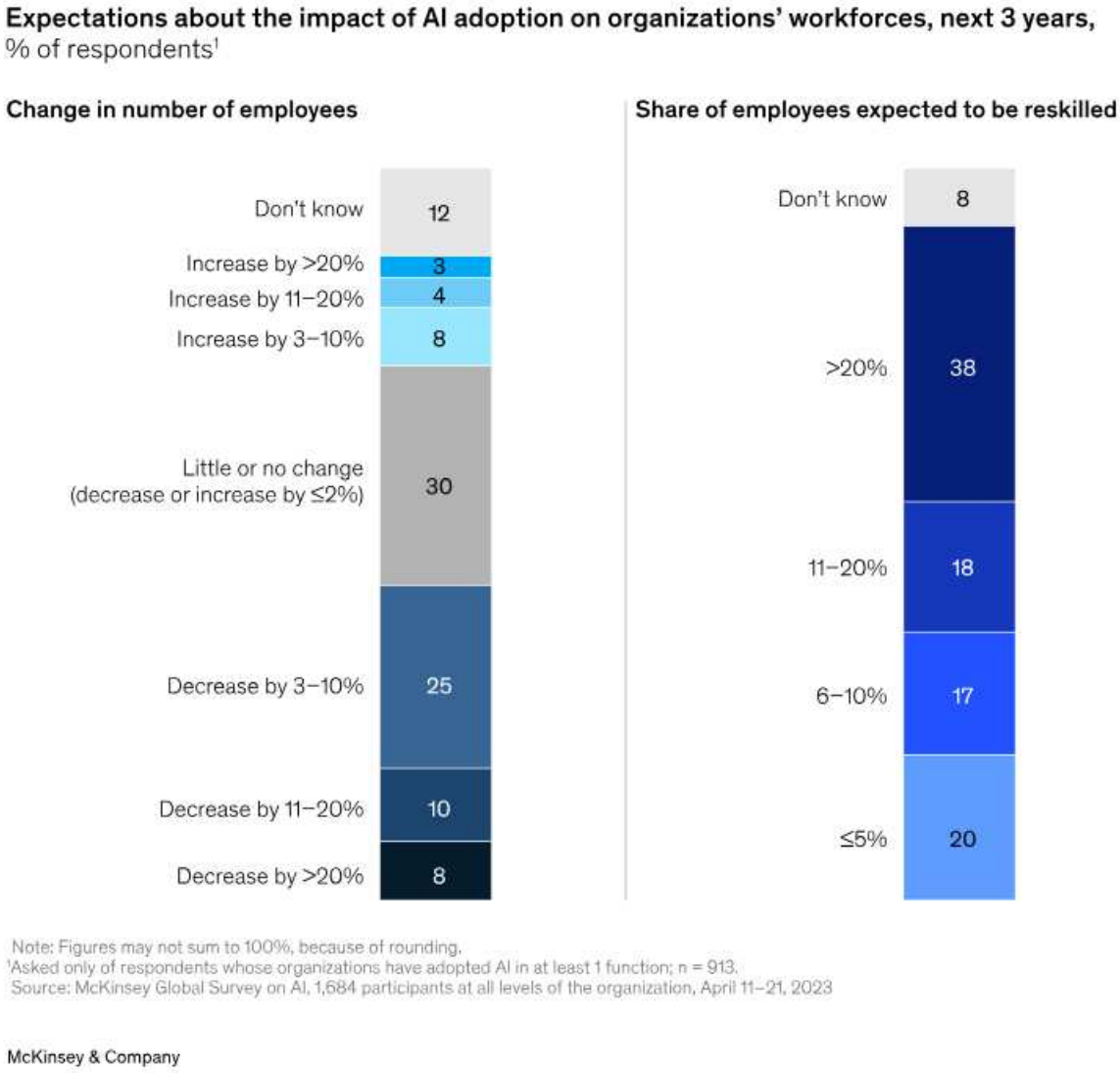


Figura 2 – Expectativas sobre o impacto da adoção de IA's.

A IA, em sua essência, de acordo com a definição mais recente de Stuart Russell, é descrita como o estudo de agentes inteligentes que podem perceber seu ambiente e tomar decisões para alcançar objetivos de forma autônoma. Estes agentes operam em contextos dinâmicos e incertos, ajustando-se ao ambiente para otimizar suas ações. A IA busca construir sistemas que possam agir de maneira racional, onde a "racionalidade"é definida pela capacidade de tomar decisões corretas, maximizando o sucesso com base nas informações disponíveis (RUSSELL; NORVIG, 2022). No entanto, quando lidamos com

dados descentralizados, como os provenientes de dispositivos pessoais, surge um dilema: como treinar modelos eficazes sem comprometer a privacidade dos usuários?

O termo "privacidade de dados"ganhou destaque no campo de tecnologias em resposta à crescente preocupação sobre como as informações pessoais dos indivíduos são coletadas, armazenadas, processadas e compartilhadas. Historicamente, o conceito de privacidade sempre esteve presente, mas a explosão no volume de dados digitais, especialmente com o advento da internet e das redes sociais, trouxe à tona desafios em relação à segurança e à confidencialidade das informações pessoais.

O marco regulatório, como a Diretiva de Proteção de Dados Pessoais da União Europeia (GDPR) (UNION, 2016), promulgada em 2016, evidencia a crescente importância atribuída à privacidade de dados em nível global. A GDPR estabeleceu normas rigorosas para a coleta e o processamento de dados pessoais, visando proporcionar aos indivíduos um maior controle sobre suas informações pessoais.

O conceito de privacidade no treinamento de redes neurais é amplamente explorado em pesquisas que buscam conciliar a necessidade de dados para treinamento eficaz com a preservação da confidencialidade. Métodos como a privacidade diferencial, proposta por Cynthia Dwork em seu trabalho *"Calibrating Noise to Sensitivity in Private Data Analysis"*(DWORK et al., 2006), introduzem ruído controlado nos dados, garantindo que as contribuições individuais não possam ser identificadas, protegendo assim a privacidade dos participantes no treinamento colaborativo.

Além disso, o aprendizado federado, proposto por McMahan(MCMAHAN et al., 2017), é uma abordagem que permite o treinamento de modelos em dados distribuídos, mantendo os dados localmente nos dispositivos dos usuários.

O aprendizado federado, do inglês federated learning (FL) foi primeiramente descrito por Brendan McMahan como o trecho de tradução própria:

> "Nós nomeamos nossa abordagem como Federated Learning, uma vez que a tarefa de aprendizagem é resolvida por uma ampla federação de dispositivos participantes (que chamamos de clientes) que são coordenados por um servidor central (MCMAHAN et al., 2017)."

Aprendizado de Máquina (AM), conforme descrito por (RUSSELL; NORVIG, 2022) e (**??**), refere-se ao campo da inteligência artificial que estuda algoritmos capazes de aprender padrões e tomar decisões com base em dados. A ideia central é que, ao invés de serem explicitamente programados para realizar uma tarefa, os modelos de AM ajustam seus parâmetros com base em grandes conjuntos de dados, de modo a gerar previsões ou classificar informações de maneira autônoma. O objetivo é permitir que as máquinas melhorem seu desempenho à medida que recebem mais informações e exemplos.

A proposta do aprendizado federado no campo de *Machine Learning* (ML) tem suas raízes em questões de privacidade de dados e descentralização. O conceito começou a ganhar destaque no final da década de 2010 como uma abordagem para treinar modelos de ML em dados distribuídos, sem a necessidade de centralizar todos os dados em um único local. Resumindo, FL é um paradigma de aprendizado onde modelos estatísticos são treinados em um sistema distribuído (LI; SHARMA; MOHANTY, 2020).

Em aplicações de FL no mundo real, as partes individuais necessitam de motivação para utilizarem um sistema de aprendizado federado (FLS), essa motivação pode vir de regulamentações ou incentivos. Quem decide utilizar o FL pode se beneficar com seu modelo de alta precisão(LI et al., 2021). Por exemplo, hospitais durante a pandemia utilizaram um sistema de ML , treinado utilizando o aprendizado federado, para classificar imagens de torax diferentes fontes (raio-x e ultrassom) com o intuíto de auxiliar no diagnóstico de COVID-19(QAYYUM et al., 2022).

Treinar modelos em redes heterogêneas e de grande escala trazem consigo uma nova gama de desafios que requerem uma abordagem fundamentalmente diferente das abordagens tradicionais de treinamento para aprendizado de máquina em larga escala (LI et al., 2020). Abordaremos durante o capítulo de fundamentação teórica os principais desafios que esse novo paradigma traz consigo, seus benefícios e o que está sendo estudado da área atualmente.

1.2 Justificativa

A necessidade de treinar modelos de inteligência artificial sem comprometer a privacidade dos usuários tem impulsionado a pesquisa em direção ao paradigma do aprendizado federado. A dependência da representatividade dos dados locais, a possível exposição de informações sensíveis durante as atualizações do modelo e a necessidade de garantir a segurança contra ataques são aspectos cruciais a serem abordados.

A diversidade nos conjuntos de dados entre os dispositivos participantes pode levar a modelos globais que não capturam adequadamente as nuances específicas de cada local. A representatividade dos dados locais é discutida em estudos como (MCMAHAN et al., 2017), que ressaltam a importância de estratégias eficientes de agregação para preservar a qualidade do modelo global.

A preservação da privacidade durante o treinamento federado é um ponto crítico, considerando a sensibilidade dos dados envolvidos. Técnicas como privacidade diferencial, discutidas em (SHOKRI; SHMATIKOV, 2015), são exploradas para mitigar o risco de divulgação de informações pessoais durante a colaboração no treinamento. No entanto, a aplicação bem-sucedida dessas técnicas requer uma análise cuidadosa das configurações e parâmetros para equilibrar a privacidade e a utilidade do modelo resultante.

A proteção contra ataques que buscam extrair informações dos modelos treinados, é outra dimensão crítica para garantir a segurança. Estudos como "Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning"(HITAJ, 2017) destacam a necessidade de estratégias defensivas robustas para preservar a privacidade contra tentativas maliciosas.

A crescente integração de inteligência artificial em nosso cotidiano, especialmente em setores como saúde, finanças e tecnologia, levanta sérias preocupações sobre a privacidade dos dados dos usuários. Incidentes recentes demonstram como a centralização de dados pode resultar em vulnerabilidades significativas. Por exemplo, em 2018, a revelação de que dados de localização de militares dos EUA foram expostos publicamente através de aplicativos de fitness gerou preocupações sobre a segurança nacional (BBC, 2018).

Além disso, a coleta indiscriminada de dados por assistentes virtuais, como o caso da Amazon Alexa, trouxe à tona a possibilidade de acesso não autorizado e uso indevido dessas informações pessoais (BLOOMBERG, 2019). Escândalos envolvendo redes sociais, como o Facebook e a Cambridge Analytica, ressaltam como a centralização de dados pode ser explorada para influenciar eventos políticos e comprometer a privacidade individual (GUARDIAN, 2018). Instituições financeiras também não estão imunes, como evidenciado pelo caso de violação de dados do Capital One(ONE, 2019).

Esses episódios enfatizam a necessidade de abordagens inovadoras para proteger a privacidade dos usuários e mitigar riscos associados à concentração de dados. O aprendizado federado surge como uma solução promissora neste cenário, permitindo que modelos de machine learning sejam treinados localmente nos dispositivos dos usuários, evitando a necessidade de transferir dados sensíveis para servidores centrais. Essa abordagem descentralizada não apenas preserva a privacidade dos indivíduos, mas também fortalece a segurança dos sistemas de IA, resguardando contra possíveis explorações maliciosas.

O campo do aprendizado federado não apenas se apresenta como uma resposta pragmática às vulnerabilidades atuais, mas também sinaliza um caminho crítico para a construção de sistemas de IA éticos e seguros em um mundo cada vez mais interconectado.

1.3 Questões de Pesquisa e Desenvolvimento

O presente trabalho é de viés aplicado, demandando atividades de pesquisa e desenvolvimento. Portanto, ao final do trabalho, pretende-se responder os seguintes questionamentos:

Questão de Pesquisa: Como o Aprendizado Federado se destaca como uma solução eficaz para o treinamento de modelos de Inteligência Artificial em ambientes descentralizados, e de que maneira ele preserva a privacidade dos usuários durante esse

processo?

Diante de estudos preliminares já realizados até o momento, e dada a relevância que a área de Inteligência Artificial vem ganhando nos últimos anos, surge uma crescente preocupacação a respeito de como são distribuídos os dados para treinamento dos modelos. O objetivo deste estudo é explorar e analisar a aplicabilidade do modelo de treinamento do aprendizado federado para a preservação e manutenção da privacidade de dados dos usuários. Demais informações sobre o tema serão apresentadas no Capítulo 2 - Referencial Teórico.

Questão de Desenvolvimento: Quais são os desafios e limitações associados à implementação prática do AF em cenários descentralizados, especialmente em termos de eficiência computacional, comunicação entre dispositivos e adaptação a diferentes tipos de dados?

A implementação prática do Aprendizado Federado em cenários descentralizados enfrenta desafios que afetam sua eficiência computacional, a comunicação entre dispositivos e a adaptação a diferentes tipos de dados. Primeiramente, a heterogeneidade de dispositivos e redes pode resultar em variações significativas na eficiência do treinamento. A comunicação entre dispositivos é limitada por questões de largura de banda e latência, impactando diretamente a sincronização eficiente dos modelos federados. Adicionalmente, a diversidade nos tipos de dados distribuídos impõe desafios na harmonização e adaptação de modelos para garantir a generalização adequada.

Neste sentido, há uma necessidade de um estudo mais criterioso sobre como desenvolver um modelo que atenda a estas preocupações. O desenvolvimento busca acompanhar e compreender como o AF pode atuar como uma solução para o treinamento descentralizado de modelos de inteligência artificial, realizando de maneira efetiva o processamento dos dados e mantendo a privacidade dos usuários.

1.4 Objetivos

A proposta desse trabalho visa explorar e analisar a aplicabilidade do Aprendizado Federado no contexto do treinamento de modelos de Inteligência Artificial em ambientes descentralizados. Este estudo é motivado pela crescente importância de preservar a privacidade dos usuários em um cenário em que dados sensíveis estão distribuídos em dispositivos diversos, como dispositivos móveis, sensores IoT e servidores locais.

O diferencial desta pesquisa reside na abordagem abrangente e aprofundada do AF em ambientes descentralizados, focando não apenas nos aspectos técnicos, mas também nas implicações éticas e práticas associadas. Ao considerar a heterogeneidade de dados, desafios de comunicação, e a necessidade crítica de proteger informações sensíveis, este

estudo visa oferecer insights valiosos para o desenvolvimento e implementação eficaz de sistemas de IA federados.

A importância deste trabalho é evidenciada pela urgência de encontrar soluções que conciliem os avanços em IA com a proteção da privacidade individual. À medida que a coleta e o processamento de dados descentralizados se tornam a norma, a comunidade científica e a indústria enfrentam o desafio de assegurar que os modelos de IA derivados desses dados não comprometam a privacidade dos usuários. Neste contexto, o AF surge como uma abordagem promissora, e este trabalho busca contribuir para a compreensão profunda de suas implicações, desafios e melhores práticas.

As contribuições deste estudo para a área de IA são diversas. Primeiramente, espera-se fornecer diretrizes práticas para a implementação bem-sucedida do AF em ambientes descentralizados, considerando os aspectos práticos e éticos. Além disso, ao abordar questões de segurança e conformidade com regulamentações de privacidade, este trabalho busca fortalecer a confiança na aplicação do AF. Por fim, ao delinear as possíveis evoluções futuras do AF e seu papel na preservação da privacidade, este estudo visa inspirar pesquisas subsequentes e avanços na área.

Neste contexto, o **Objetivo geral** é investigar como o treinamento de inteligências artificiais por meio do aprendizado federado pode ser efetivamente aplicado no contexto de dados descentralizados.

Com o norte do objetivo geral e sua abrangência prevista, pretende-se atingi-lo através do cumprimimento de alguns **Objetivos específicos** sendo eles:

- Objetivo específico 1: Conceituar Aprendizado federado e apontar as estratégias de aprendizado de máquina que podem ser usadas para preservação da privacidade dos usuários e de dados sensíveis.
- Objetivo específico 2: Discorrer sobre técnicas de segurança e privacidade necessárias para proteger os dados dos usuários e o provedor do serviço durante o processo de treinamento.
- Objetivo específico 3: Analisar o problema de uso de dados decentralizados, identificando os principais desafios.
- Objetivo específico 4: Comparar a eficácia do modelo de treinamento descentralizado quando comparado com modelos tradicionais de treinamento.

É importante salientar que, para cumprimento desses objetivos, sobre a necessidade de orientação por metodologias de:

- pesquisa, para condução dos estudos e levantamentos;

- desenvolvimento, para construção da solução em si, e
- análise de resultados, para lidar mais adequadamente com os resultados obtidos.

.

1.5 Materiais e Métodos

No capítulo de materiais e métodos, serão tratados os aspectos metodológicos do trabalho de forma mais adequada. Entretanto, visando esclarecer alguns pontos, o trabalho pode ser classificado segundo (GIL, 2002), quanto:

- **Quanto aos objetivos**: a pesquisa é classificada como exploratória, pois busca explorar o tema do aprendizado federado e sua aplicabilidade no treinamento de modelos de IA em ambientes descentralizados, bem como investigar os desafios e limitações associados à implementação prática do AF.
- **Quanto aos procedimentos técnicos**: a pesquisa é classificada como bibliográfica-experimental, pois envolve a revisão de literatura sobre aprendizado federado, privacidade de dados, segurança em IA e tópicos relacionados assim como o desenvolvimento de modelos que servirão como objetos de estudo para definição de formas de controle e observação dos efeitos que a variável produz no objeto.
- **Quanto à abordagem do problema**: a pesquisa é classificada como Quali-quantitativas, pois busca compreender e analisar os aspectos práticos e éticos do AF em ambientes descentralizados, considerando a heterogeneidade de dados, desafios de comunicação e a necessidade de proteger informações sensíveism, combinando a análise de métricas e números gerados com os experimentos.

1.6 Organização

A monografia está organizada em capítulos, conforme a seguir:

- Capítulo 1 - Fundamentação Teorica: Neste capítulo, são apresentados os conceitos fundamentais relacionados ao aprendizado federado, privacidade de dados, segurança em IA e tópicos relacionados. O objetivo é fornecer uma base teórica sólida para a compreensão do tema e a análise dos resultados.
- Capítulo 2 - Referencial Tecnológico: descreve as principais tecnologias, ferramentas e frameworks, ou seja, o arcabouço tecnológico que viabilizou não apenas o desenvolvimento da solução em si, mas também seu planejamento, sua especificação e sua análise;

- Capítulo 3 - Materiais e métodos: descreve os procedimentos técnicos e metodológicos adotados para a realização do trabalho, incluindo a revisão bibliográfica, a coleta de dados, a análise dos resultados e a avaliação da solução proposta;
- Capítulo 4 - Desenvolvimento: descreve o desenvolvimento da solução proposta, a implementação dos modelos de IA, a integração com as tecnologias selecionadas e os resultados obtidos;
- Capítulo 5 - Análise de Resultados: apresenta a análise dos resultados obtidos, a comparação entre os modelos de IA treinados e a discussão sobre os desafios e limitações identificados;
- Capítulo 6 - Conclusão: apresenta as conclusões finais do trabalho, destacando as contribuições, as limitações e as sugestões para trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo, são conferidos os principais referenciais que embasam o presente trabalho, com o intuito de realizar contribuições para a área de Engenharia de *Software*, especificamente ao que diz respeito as áreas de Inteligencia artificial e Segurança de dados.

O Capítulo encontra-se organizado em seções, sendo as mesmas escritas orientando-se pela literatura: Fundamentos de Aprendizado de Máquina, focando na contextualização do que é machine learning e seus algoritmos de treinamento tradicionais. Em sequência abordaremos o principal tópico de estudo, o tópico de Aprendizado federado vai tratar do surgimento desse tipo de treinamento, algumas das oportunidades e desafios que o modelo de treinamento enfrenta, ferramentas de proteção de dados associadas ao modelo, tipos de ataques e medidas de segurança em ambientes descentralizados, alguns estudos de caso e aplicações práticas e por último, desenvolvimentos tecnológicos e perspectivas futuras para o Aprendizado federado.

2.1 Fundamentos de Aprendizado de Máquina

2.1.1 Definição

O campo do aprendizado de máquina, uma disciplina essencial dentro da inteligência artificial, tem raízes históricas profundas. A criação do termo "aprendizado de máquina"é creditado a Arthur Samuel, um pioneiro no campo, que o introduziu em 1959(SAMUEL, 1959). Samuel, conhecido por suas contribuições inovadoras no desenvolvimento de programas de xadrez autônomos, formalizou o conceito como a habilidade de sistemas computacionais aprenderem automaticamente sem programação explícita. A visão de Samuel estabeleceu as bases para a revolução na interação entre máquinas e dados, antecipando a evolução de algoritmos capazes de aprimorar seu desempenho ao longo do tempo por meio da adaptação a novas informações e experiências.

Durante os anos 80 e 90, a ênfase se deslocou para modelos mais complexos, incluindo redes neurais, com pesquisadores notáveis como Geoffrey Hinton, Yann LeCun e Yoshua Bengio desempenhando papéis cruciais. Essa época marcou a adoção crescente de redes neurais profundas e técnicas de aprendizado profundo, transformando o aprendizado de máquina e inaugurando uma nova era (HINTON et al., 2012). O campo continuou a evoluir para abranger técnicas diversas, desde aprendizado supervisionado e não supervisionado até métodos mais avançados, como aprendizado por reforço e aprendizado semi-supervisionado abordados em livros como "Deep Learning"escrito por Goodfellow (GOODFELLOW, 2016).

A definição contemporânea de aprendizado de máquina engloba uma diversidade de técnicas que se adaptam às demandas crescentes de manipulação de grandes volumes de dados e à necessidade de extrair insights, não se limitando mais apenas a algoritmos preditivos. Além disso, lidar com problemas atuais, como ética e privacidade, mostra que é importante equilibrar a inovação com questões sociais e éticas como abordado por Floridi e Cowls(FLORIDI; COWLS, 2018).

2.1.2 Algoritmos de Treinamento Tradicionais

Algoritmos clássicos têm desempenhado um papel crucial no progresso notável alcançado em aplicações de inteligência artificial, alguns deles como o de Regressão Linear na previsão de tendências econômicas, Máquinas de Vetores de Suporte (SVM) para diagnósticos médicos para classificação de imagens, ajudando na detecção de câncer em imagens de ressonância magnética ou tomografia computadorizada, e Redes Neurais de Feedforward em sistemas de reconhecimento de fala. A robustez desses métodos reside na simplicidade conceitual e na interpretabilidade, sendo capazes de aprender padrões a partir de dados de treinamento para realizar tarefas específicas. A Regressão Linear, por exemplo, é amplamente utilizada em problemas de previsão, enquanto as SVMs têm se destacado em tarefas de classificação, e as Redes Neurais de Feedforward têm demonstrado eficácia em problemas complexos, proporcionando um entendimento mais profundo dos dados (BISHOP, 2006) (RUSSELL; NORVIG, 2016).

Um dos desafios inerentes é a capacidade de lidar com dados não lineares de forma eficiente. Enquanto métodos como SVM podem ser estendidos para incluir kernels não lineares, a complexidade computacional muitas vezes aumenta substancialmente. Além disso, a interpretabilidade pode diminuir à medida que a complexidade do modelo aumenta, tornando difícil compreender os processos subjacentes, especialmente em redes neurais profundas (GOH, 2017).

Quando se trata de privacidade de dados, esses algoritmos tradicionais enfrentam algumas dificuldades. A coleta centralizada de dados para treinamento pode comprometer a privacidade dos usuários, expondo informações sensíveis a potenciais violações de segurança. Métodos como a Regressão Logística podem ser vulneráveis a ataques de reengenharia, nos quais modelos treinados são explorados para recuperar dados originais (FREDRIKSON; JHA; RISTENPART, 2014).

A constante busca por algoritmos mais eficientes e seguros destaca a importância de equilibrar os pontos fortes e fracos dessas abordagens tradicionais. À medida que avançamos, é crucial incorporar técnicas inovadoras, como o aprendizado federado e a privacidade diferencial, para superar os desafios relacionados à privacidade de dados e aprimorar a eficácia geral dos modelos de inteligência artificial.

2.2 Aprendizado Federado

2.2.1 Definição

O termo "Aprendizado Federado"foi introduzido por Mcmahan em 2016, marcando uma mudança paradigmática na colaboração descentralizada de modelos de inteligência artificial(MCMAHAN et al., 2017). Essa abordagem, muitas vezes referida como "Aprendizado Colaborativo", destaca-se por permitir que modelos sejam treinados localmente em dispositivos distribuídos, preservando a privacidade dos dados dos usuários enquanto contribui para um modelo global aprimorado.

Os pontos fortes do AF residem na capacidade de treinar modelos de maneira colaborativa, alavancando a diversidade de dados locais sem a necessidade de compartilhar informações sensíveis centralmente. Isso se revela especialmente valioso em cenários onde a segurança e a privacidade dos dados são prioridades, como é o caso em ambientes de saúde e finanças. A descentralização do treinamento permite que organizações colaborem efetivamente sem comprometer a confidencialidade dos dados(YANG et al., 2019).

Quando se trata de privacidade de dados, o Aprendizado Federado enfrenta desafios singulares. Embora a descentralização do treinamento preserve a privacidade local, informações sensíveis ainda podem ser inferidas a partir dos modelos globais. Técnicas avançadas, como a aplicação de Privacidade Diferencial, tornam-se essenciais para mitigar riscos de divulgação de informações individuais durante a colaboração federada(GEYER et al., 2017).

É interessante explorar as complicações do AF em comparação com os métodos tradicionais de treinamento de modelos de inteligência artificial. No treinamento convencional, um modelo centralizado é alimentado com dados provenientes de diversas fontes, consolidando-os em um único local para o ajuste dos parâmetros. Esse processo, embora eficaz, levanta preocupações quanto à privacidade, segurança e eficiência, especialmente quando lidamos com dados sensíveis e distribuídos (GOODFELLOW, 2016).

No treinamento tradicional, os dados são reunidos em um servidor central, onde o modelo é ajustado iterativamente utilizando a metodologia de otimização de gradiente descendente. Esse método, embora tenha sido a base para muitos avanços na inteligência artificial, enfrenta algumas dificuldades como a necessidade de compartilhamento massivo de dados, o que pode comprometer a privacidade do usuário. A centralização dos dados também implica em vulnerabilidades de segurança, tornando-se um ponto único de falha. Adicionalmente, o treinamento tradicional pode tornar-se computacionalmente oneroso, especialmente em grandes conjuntos de dados e modelos complexos (SUTSKEVER et al., 2013).

O Aprendizado Federado, por outro lado, propõe uma abordagem descentralizada

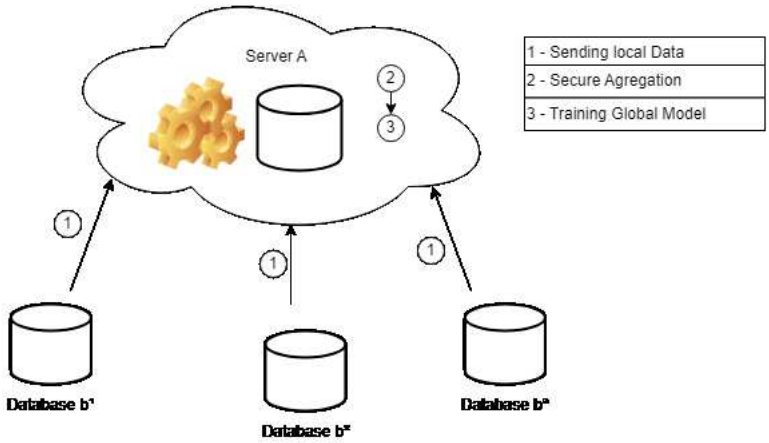


Figura 3 – Modelo de Treinamento Centralizado.

e colaborativa. No contexto do AF, os modelos são treinados localmente em dispositivos distribuídos, como smartphones ou servidores locais, antes de compartilhar atualizações resumidas com um servidor central. Este, por sua vez, coordena as contribuições de todos os dispositivos para ajustar o modelo global. Esse processo de treinamento colaborativo permite a preservação da privacidade dos dados, uma vez que informações sensíveis permanecem no local de origem(MCMAHAN et al., 2017).

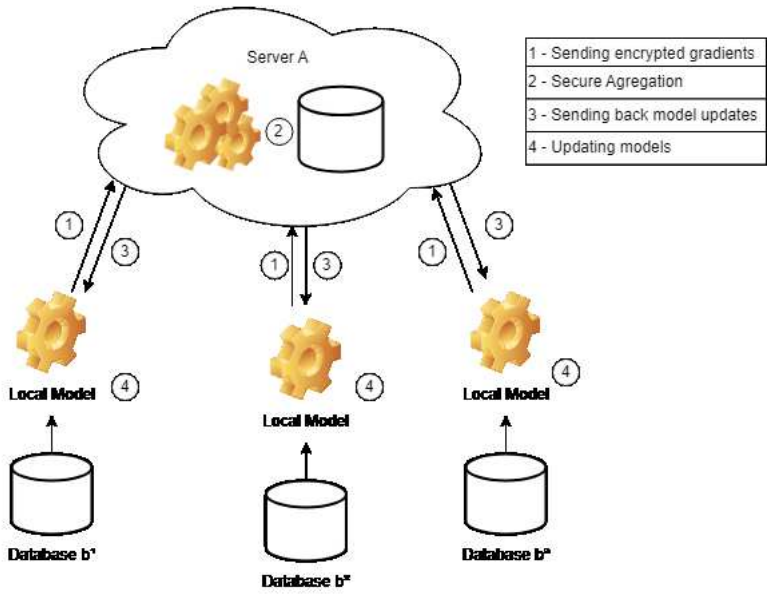


Figura 4 – Modelo de Treinamento - Aprendizado Federado.

A eficácia do Aprendizado Federado é evidenciada pela sua capacidade de superar algumas das limitações dos métodos tradicionais. Ao descentralizar o treinamento, o AF elimina a necessidade de transferir grandes volumes de dados, facilitando a mitigação dos riscos associados à exposição de informações sensíveis. Além disso, o Aprendizado Federado oferece uma solução eficaz para a adaptação de modelos a diferentes características

dos dados locais, promovendo a robustez e generalização do modelo global. A descentralização do treinamento reduz a carga computacional no servidor central, distribuindo o esforço computacional entre os dispositivos participantes(YANG et al., 2019). Isso não apenas acelera o treinamento, mas também possibilita a utilização de dispositivos com recursos computacionais limitados, tornando o processo mais eficiente e acessível. Além disso, a abordagem federada minimiza os riscos de segurança, pois não há necessidade de transferir dados sensíveis centralmente(BONAWITZ et al., 2019).

2.2.2 Oportunidades e Desafios

O surgimento do aprendizado federado vem para aproveitar das principais fragilidades dos modelos de treinamento com dados centralizados como a capacidade de preservar a privacidade dos usuários ou a heterogeneidade dos dados. Ao permitir que o treinamento ocorra localmente em dispositivos individuais, o AF elimina a necessidade de transferir grandes volumes de dados para um servidor central, mitigando assim os riscos associados à exposição de informações sensíveis. Essa abordagem é particularmente relevante em setores como saúde e finanças, onde a confidencialidade dos dados é imperativa (MCMAHAN et al., 2017).

Além disso, o Aprendizado Federado oferece uma solução para o desafio da heterogeneidade dos dados distribuídos. Ao permitir que modelos sejam treinados em locais distintos, onde as características dos dados podem variar, o AF promove a adaptação local sem comprometer a criação de um modelo global robusto. Essa flexibilidade é crucial em cenários nos quais a diversidade de dados é uma característica intrínseca, como em aplicações urbanas ou de Internet das Coisas (IOT) (YANG et al., 2019).

Ao descentralizar o processo de treinamento, o AF reduz a carga computacional em um servidor central, distribuindo o esforço computacional entre os dispositivos participantes. Isso não apenas acelera o treinamento, mas também possibilita a utilização de dispositivos com recursos computacionais limitados, ampliando o alcance do treinamento de modelos de inteligência artificial(BONAWITZ et al., 2019).

2.2.3 Privacidade Diferencial como Ferramenta de Proteção

Desenvolvida para proteger informações sensíveis durante o processamento de dados, a Privacidade Diferencial (DP) oferece uma abordagem robusta e matematicamente fundamentada para garantir a confidencialidade dos dados individuais (DWORK, 2011). A Privacidade Diferencial baseia-se no princípio de adicionar ruído controlado aos resultados de consultas realizadas em um banco de dados. Esse ruído introduzido protege a privacidade individual, tornando difícil discernir a contribuição específica de um único dado no conjunto. Formalmente, um mecanismo é diferencialmente privado se, para qual-

quer dois conjuntos de dados que diferem apenas em um elemento, a probabilidade de obter o mesmo resultado permanece próxima, mesmo que seja revelado o resultado da consulta (DWORK, 2011).

No cenário do AF, a integração da Privacidade Diferencial é essencial para assegurar a confidencialidade dos dados contribuídos por dispositivos locais. O processo de treinamento federado envolve a colaboração de modelos locais em dispositivos distribuídos, mas a agregação dessas informações pode expor informações sensíveis. Ao aplicar mecanismos de DP durante a agregação, cada dispositivo contribui diferencialmente para o modelo global, mitigando riscos de divulgação de dados individuais(ABADI; AL., 2016).

Para ilustrar a eficácia da Privacidade Diferencial no treinamento federado, consideremos o exemplo de um sistema de recomendação personalizada. Ao incorporar técnicas de DP na agregação de preferências locais dos usuários, o sistema pode aprender padrões de comportamento de forma coletiva sem expor escolhas individuais. Da mesma forma, em diagnósticos médicos distribuídos, a aplicação de DP assegura que os modelos federados aprendam com imagens sem revelar informações específicas de pacientes, promovendo avanços na pesquisa médica colaborativa(SHOKRI; AL., 2015).

2.2.4 Ataques e medidas de Segurança em Ambientes Descentralizados

Embora o AF ofereça uma solução promissora para a preservação da privacidade dos dados, ele também enfrenta ameaças consideráveis provenientes de ataques sofisticados. A compreensão desses desafios é essencial para avançar em direção a modelos mais seguros e robustos(GEYER et al., 2017). Alguns dos ataques mais comuns utilizados contra os modelos de treinamento descentralizados são os:

- Ataques de Inversão de Modelo
- Ataques de Envenenamento de Dados
- Ataques de Inferência Estatística

2.2.4.1 Ataques de Inversão de Modelo

Um dos principais desafios de segurança em modelos descentralizados é representado pelos ataques de inversão de modelo. Nestes ataques, um adversário tenta inferir informações sensíveis sobre os dados de treinamento a partir do modelo global treinado. Essa técnica pode ser particularmente eficaz quando o modelo é treinado em dados sensíveis, como informações médicas ou financeiras. Estratégias avançadas, como a reconstrução do conjunto de treinamento original, podem ser empregadas para recuperar detalhes específicos, comprometendo assim a privacidade dos dados(FREDRIKSON; JHA; RISTENPART, 2014). Para mitigar os ataques de inversão de modelo em treinamentos descentralizados,

estratégias como **limitação de informações no modelo global** (Reduzir a quantidade de informações específicas incorporadas no modelo global pode minimizar o risco de ataques) e **Controle de acessos**, podem ser utilizadas.

2.2.4.2 Ataques de Envenenamento de Dados

Outra classe crítica de ataques em modelos descentralizados é representada pelos ataques de envenenamento de dados. Nestes cenários, um adversário introduz intencionalmente dados maliciosos no conjunto de treinamento distribuído, visando manipular o modelo global resultante. Esses ataques podem levar a decisões incorretas do modelo e comprometer sua utilidade e confiabilidade. Estratégias de defesa, como **detecção de dados envenenados** e **técnicas de aprendizado robusto**, tornam-se essenciais para mitigar esses riscos(BAGDASARYAN; VEJDEMO-JOHANSSON; SHMATIKOV, 2018).

2.2.4.3 Ataques de Inferência Estatística

Além disso, modelos de treinamento descentralizados estão sujeitos a ataques de inferência estatística, nos quais adversários tentam deduzir informações sensíveis explorando padrões estatísticos nos modelos ou nas atualizações de parâmetros. Esses ataques podem revelar informações sobre os dados de treinamento local, mesmo sem acesso direto a eles. Técnicas como **Privacidade Diferencial** têm sido propostas para mitigar essas ameaças, introduzindo ruídos controlados nas atualizações de modelo tornando mais difícil para um atacante inferir informações específicas sobre os dados locais(SHOKRI; SHMATIKOV, 2015).

2.2.5 Estudos de Caso e Aplicações Práticas

O crescente volume de dados e a necessidade de preservar a privacidade dos usuários impulsionaram o desenvolvimento do Aprendizado Federado como uma abordagem no treinamento de modelos de Inteligência Artificial em ambientes descentralizados. Diversos estudos de caso destacam o sucesso do AF em diferentes contextos. Em um desses casos, a aplicação do AF na área de saúde permitiu o treinamento de modelos de diagnóstico em dados distribuídos entre várias instituições médicas, preservando informações sensíveis dos pacientes (SMITH; AL., 2018). Em outro cenário, empresas financeiras adotaram o AF para a detecção de fraudes, permitindo a colaboração entre instituições sem a necessidade de compartilhar detalhes específicos de transações individuais(JONES; AL., 2020).

Outro estudo de caso na área de educação revela como o AF está transformando o cenário educacional. Em uma colaboração entre instituições acadêmicas, o modelo de treinamento foi implementado para aprimorar a personalização do ensino sem comprometer a privacidade dos estudantes (EduTechCollab, 2020). Os modelos de IA treinados localmente em diferentes escolas foram agregados de forma segura, permitindo que os

educadores recebessem insights valiosos sobre o desempenho dos alunos, adaptando estratégias de ensino de maneira eficaz. Esse estudo não apenas demonstrou a eficácia do AF na área educacional, mas também destacou sua capacidade de superar desafios específicos relacionados à privacidade e à diversidade dos dados.

Além dos setores tradicionais, o Aprendizado Federado tem se destacado na promoção da colaboração científica. Em um estudo colaborativo entre instituições de pesquisa em diferentes países, o AF foi utilizado para treinar modelos de predição em conjuntos de dados descentralizados, envolvendo pesquisadores em diversas disciplinas (SciCollab, 2021). O estudo evidenciou não apenas a aplicabilidade do AF em ambientes científicos, mas também seu potencial para integrar especialidades diversas, impulsionando descobertas e avanços mais amplos na pesquisa.

Resultados obtidos em projetos relevantes reforçam a eficácia do AF. Uma pesquisa acadêmica conduzida por especialistas em privacidade e IA demonstrou que a implementação do AF em plataformas de redes sociais possibilitou a personalização de recomendações de conteúdo sem comprometer a privacidade dos usuários (PrivacyInTech, 2022). Em um projeto colaborativo entre instituições de pesquisa e empresas do setor de energia, o AF facilitou a previsão de demanda de forma descentralizada, garantindo a confidencialidade dos dados operacionais (EnergyTechCollab, 2019).

A revisão desses estudos de caso destaca a versatilidade do AF e sua aplicabilidade em diversos setores. A capacidade de treinar modelos de IA em dados descentralizados, preservando a privacidade dos usuários, é um marco na busca por soluções éticas e eficientes. Esses casos oferecem insights valiosos para pesquisadores, profissionais e gestores que buscam adotar o AF em suas práticas, promovendo uma abordagem mais colaborativa e segura no avanço da IA.

2.2.6 Perspectivas Futuras e Desenvolvimentos Tecnológicos

Uma tendência emergente crucial é a busca pela otimização do processo federado, visando aprimorar a eficiência e a eficácia do treinamento. A introdução de algoritmos de otimização mais avançados, como algoritmos federados de segunda ordem, tem o potencial de acelerar a convergência do treinamento, reduzindo a carga computacional nos dispositivos participantes e, consequentemente, promovendo uma maior adesão ao modelo federado (KONEčNý; AL., 2016).

Outra tendência que ganha destaque é a aplicação do AF em cenários de aprendizado contínuo e online. Com o crescente volume de dados gerados em tempo real, o AF está sendo adaptado para suportar treinamento incremental, permitindo que modelos federados evoluam continuamente à medida que novos dados são incorporados. Essa abordagem é particularmente relevante em ambientes como a Internet das Coisas (IoT),

onde a natureza dinâmica dos dados demanda modelos de IA que possam se adaptar rapidamente às mudanças nas condições operacionais (MOHASSEL; AL., 2017).

A evolução do AF não ocorre isoladamente, ela está intrinsecamente ligada ao progresso de tecnologias emergentes. O advento da computação quântica é uma área que pode revolucionar o treinamento federado. Algoritmos quânticos prometem realizar cálculos de forma exponencialmente mais eficiente do que os algoritmos clássicos, potencialmente reduzindo significativamente o tempo necessário para treinar modelos federados complexos (AARONSON, 2016).

A segurança no AF é uma consideração crítica e as tecnologias emergentes no campo da segurança cibernética também desempenharão um papel fundamental. A aplicação de técnicas de criptografia pós-quântica pode fortalecer ainda mais a segurança do AF, garantindo a confidencialidade dos modelos e dados distribuídos. Essa combinação de AF e criptografia pós-quântica pode proporcionar um ambiente de treinamento mais robusto em face de ameaças potenciais (GIOVANNI; AL., 2020).

3 Referencial Tecnológico

Neste capítulo, serão apresentadas as principais ferramentas e tecnologias que estão sendo utilizadas para elaboração, condução, desenvolvimento e análise do trabalho como um todo. Começando pelas principais ferramentas, as quais viabilizam o desenvolvimento e a análise do projeto, Ferramentas Associadas ao projeto, tem-se que: o versionamento/hospedagem com uso do GitHub;

Entretanto, ainda existem outros apoios tecnológicos de relevâncias, e que merecem ser destacados. Em Outros Apoios Tecnológicos, há destaque para Ferramentas de Comunicação (ex. Teams e Whatsapp); e Ferramentas de Escrita da Monografia (ex. Latex e Overleaf).

3.1 Ferramentas Associadas ao Projeto

Seguem as principais ferramentas que serão consumidas para viabilizar o desenvolvimento e a análise do projeto.

3.1.1 Python

Python é uma linguagem de programação de alto nível, interpretada e de propósito geral, conhecida por sua simplicidade e legibilidade. Criada por Guido van Rossum, Ela foi concebida com ênfase na facilidade de uso e na capacidade de expressar ideias de forma concisa. Seu design modular e sintaxe clara a torna uma escolha popular tanto para iniciantes quanto para desenvolvedores experientes. A filosofia central de Python, conhecida como "Zen of Python,"destaca princípios como clareza, simplicidade e praticidade, guiando o desenvolvimento da linguagem.

A linguagem suporta diversos paradigmas de programação, sendo mais notavelmente associado à programação orientada a objetos. No entanto, a linguagem é multifacetada, permitindo abordagens imperativas, funcionais e procedural. Essa flexibilidade permite aos desenvolvedores escolherem o paradigma que melhor se adapta ao problema em questão, contribuindo para a versatilidade da linguagem em diferentes contextos de desenvolvimento.

Em relação aos seus prós, Python é reconhecido pela sua legibilidade e sintaxe limpa, fatores que facilitam a manutenção e colaboração em projetos. A vasta biblioteca padrão e a extensa comunidade de desenvolvedores oferecem suporte para uma ampla gama de tarefas, desde desenvolvimento web até análise de dados.

A sintaxe clara e a legibilidade do Python tornam o desenvolvimento de algoritmos de IA mais acessível, permitindo que os desenvolvedores foquem na lógica e nos conceitos subjacentes, em vez de se perderem em detalhes complexos de implementação. Além disso, a vasta biblioteca padrão e o ecossistema robusto de bibliotecas de terceiros, como NumPy, Pandas e Scikit-learn, oferecem ferramentas poderosas para manipulação de dados, processamento matemático e implementação eficiente de algoritmos de aprendizado de máquina.

Outro ponto crucial é a popularidade e o suporte contínuo que Python recebe na comunidade de ciência de dados e IA. A abundância de frameworks e bibliotecas especializadas, como TensorFlow, PyTorch e scikit-learn, todos com interfaces amigáveis, consolida a posição da linguagem como uma escolha estratégica para desenvolvedores e pesquisadores em IA. A combinação de acessibilidade, riqueza de ferramentas e apoio da comunidade faz de Python uma linguagem ideal para a concepção, prototipagem e implementação eficiente de soluções de inteligência artificial.

3.1.2 TensorFlow

TensorFlow é uma poderosa biblioteca de código aberto desenvolvida pelo Google, projetada para facilitar a implementação de algoritmos de aprendizado de máquina e redes neurais. Destaca-se como uma das ferramentas mais populares e abrangentes no campo da inteligência artificial (IA) e aprendizado de máquina (ML). Seu nome deriva da estrutura subjacente de dados que representa fluxos multidimensionais, essenciais para operações matemáticas eficientes, fundamentais no contexto do aprendizado profundo.

A principal força do TensorFlow reside em sua flexibilidade e suporte para diversas plataformas, possibilitando o desenvolvimento e treinamento de modelos em diferentes ambientes, desde dispositivos móveis até sistemas distribuídos em larga escala. Adotando uma abordagem simbólica, o TensorFlow permite que os desenvolvedores definam e otimizem modelos complexos de maneira eficiente, com ênfase especial em redes neurais profundas. Essa flexibilidade é complementada por uma comunidade ativa e uma vasta documentação, facilitando a curva de aprendizado para novos usuários.

Entre os pontos positivos do TensorFlow, destaca-se a sua escalabilidade e eficácia no treinamento de modelos em grandes conjuntos de dados. Sua integração com unidades de processamento gráfico (GPUs) e unidades de processamento tensorial (TPUs) acelera o processamento, tornando-o ideal para projetos de aprendizado profundo.

3.1.3 Keras

Keras é uma poderosa API de alto nível projetada para simplificar o desenvolvimento de modelos de redes neurais, integrando-se perfeitamente ao TensorFlow, sendo

frequentemente utilizada em conjunto com essa biblioteca. Criada com o objetivo de tornar o desenvolvimento de modelos de aprendizado profundo mais acessível, Keras oferece uma interface intuitiva e amigável, que abstrai grande parte da complexidade inerente ao aprendizado de máquina, permitindo que desenvolvedores e pesquisadores criem, treinem e testem modelos de forma rápida e eficiente.

Keras desempenha um papel de facilitar a criação e a customização de redes neurais que podem ser adaptadas tanto para o treinamento centralizado quanto para o treinamento distribuído. Sua integração com o *TensorFlow Federated* permite que modelos Keras sejam aplicados em ambientes federados, onde o aprendizado ocorre de maneira descentralizada, mantendo a privacidade dos dados. Isso é particularmente relevante para o treinamento de modelos em sistemas de dados distribuídos, como em cenários onde os dados sensíveis são mantidos nos dispositivos dos usuários.

A simplicidade e modularidade do Keras o tornam uma ferramenta ideal para quem está começando no aprendizado de máquina, ao mesmo tempo em que oferece flexibilidade suficiente para projetos mais avançados. A API permite a construção rápida de modelos sequenciais e funcionais, suportando camadas como convolucionais, totalmente conectadas e recorrentes, as quais são essenciais para resolver problemas complexos em visão computacional, processamento de linguagem natural e outras áreas de IA.

Além disso, Keras oferece suporte para diversas otimizações e métricas que são fundamentais para a análise de desempenho dos modelos, seja em um ambiente centralizado ou federado. Sua capacidade de compilar modelos com diferentes otimizadores e funções de perda, como o *Adam* e a *sparse categorical crossentropy*, permite ajustar o comportamento dos modelos conforme as necessidades específicas do experimento, como no caso do estudo comparativo entre aprendizado centralizado e federado.

Assim como o TensorFlow, o Keras também se beneficia de uma comunidade ativa, rica documentação e suporte para execução em GPUs e TPUs, tornando-o adequado tanto para ambientes de pesquisa quanto para aplicações de produção em larga escala. Isso reforça a importância de sua escolha em projetos que buscam escalabilidade e eficiência, como os que envolvem aprendizado federado, onde o objetivo é treinar modelos em dispositivos descentralizados sem comprometer a privacidade dos dados.

3.1.4 Google Colab

O Google Colab é uma plataforma baseada em nuvem que oferece um ambiente interativo para o desenvolvimento e execução de código Python, especialmente voltado para o aprendizado de máquina e análise de dados. A ferramenta proporciona a capacidade de criar e compartilhar notebooks Jupyter, que permitem a integração de código executável, visualizações e texto descritivo em um único documento. O Google Colab é

amplamente valorizado por sua acessibilidade, uma vez que não requer configuração local e oferece acesso a recursos computacionais avançados, como unidades de processamento gráfico (GPUs) e unidades de processamento tensorial (TPUs), sem custo adicional.

No contexto deste projeto, o Google Colab foi utilizado para implementar e testar o modelo de aprendizado federado. A plataforma facilitou a execução do código de maneira eficiente e a utilização de GPUs para acelerar o treinamento do modelo, o que foi crucial para lidar com a complexidade do aprendizado federado e o processamento de grandes conjuntos de dados. Além disso, o Google Colab permitiu a documentação do progresso por meio de notebooks, que incluíram não apenas o código e os resultados das execuções, mas também as análises e visualizações das métricas de desempenho do modelo. A possibilidade de compartilhar facilmente os notebooks com os orientadores e colegas foi um aspecto importante para a colaboração e revisão contínua do projeto.

3.1.5 Base de dados Fashion-MNIST

A base de dados Fashion-MNIST foi desenvolvida como uma alternativa mais complexa ao tradicional MNIST, visando representar um cenário mais próximo de aplicações reais em visão computacional. Composta por imagens de 28x28 pixels em tons de cinza, ela contém 70.000 imagens divididas em 10 classes, todas relacionadas a itens de vestuário, como camisetas, sapatos, bolsas e casacos. Cada classe possui uma variabilidade significativa em termos de estilo e design, o que desafia os modelos de aprendizado de máquina a reconhecer padrões sutis entre os diferentes itens.

No contexto do aprendizado federado, que é o foco principal do projeto, o Fashion-MNIST oferece uma excelente oportunidade para testar a robustez de modelos distribuídos na identificação de padrões em dados não centralizados. Como os dados de usuários de dispositivos móveis, por exemplo, estão frequentemente dispersos e não podem ser agregados facilmente por questões de privacidade, a variabilidade dos dados na base Fashion-MNIST simula um ambiente similar. Cada dispositivo ou cliente pode treinar um subconjunto desses dados, representando diferentes distribuições de moda, para então colaborar no aprimoramento de um modelo global.

3.2 Outros Apoios Tecnológicos

Seguem as demais ferramentas que auxiliam na na comunicação entre os envolvidos no projeto, bem como na escrita dessa monografia.

3.2.1 Ferramentas de Comunicação

Para comunicação entre autor e orientadores, durante a elaboração desse trabalho, destacam-se:

- O Microsoft Teams, para reuniões virtuais sobre o andamento do trabalho, e
- O Whatsapp, para trocas de mensagens e avisos rápidos.

3.2.2 Ferramentas de Escrita da Monografia

Para escrita da monografia, destaca-ase o uso do LaTeX, sendo este uma linguagem de marcação para escrita de monografias comumente utilizada. Como principal vantagem tem-se a escrita em forma de texto simples, sem haver tanta preocupação com a formatação durante a escrita do texto. Adicionalmente foi utilizado o Overleaf para escrita do texto no formato LaTeX e organização das pastas, além de ser utilizado para compilar a monografia em formato PDF.

3.2.3 GitHub

O GitHub é uma plataforma amplamente utilizada para o versionamento de código e colaboração em projetos de software. Baseado no sistema de controle de versão Git, o GitHub oferece uma interface web que permite o gerenciamento de repositórios de código, acompanhamento de mudanças e coordenação entre equipes de desenvolvimento. A plataforma proporciona funcionalidades essenciais como a criação de branches para desenvolvimento paralelo, pull requests para revisão de código e issues para rastreamento de bugs e tarefas.

No contexto desta pesquisa, o GitHub foi empregado para organizar e versionar o código-fonte desenvolvido, facilitando a colaboração e a documentação do progresso do projeto. A utilização do GitHub permitiu o controle de versões do código, a integração de novas funcionalidades e a correção de eventuais falhas de forma sistemática e rastreável. Além disso, a plataforma facilitou a colaboração e comunicação com orientadores e colegas, assegurando uma gestão eficiente do desenvolvimento do projeto e garantindo a integridade e a consistência do código ao longo das diferentes etapas do trabalho.

4 Materiais e Métodos

4.1 Método adotado

Este trabalho adota uma abordagem de revisão de literatura integrativa combinada com um estudo experimental. A revisão de literatura tem como objetivo sintetizar o estado atual da pesquisa sobre aprendizado federado (Federated Learning – FL), focando em sua aplicabilidade no treinamento de modelos de IA com dados descentralizados, preservando a privacidade dos usuários. A revisão será baseada em artigos científicos, teses, dissertações e relatórios de conferências. O estudo experimental complementará a revisão, implementando um modelo de aprendizado federado utilizando o framework TensorFlow Federated (TFF) para treinar uma rede neural com o conjunto de dados Fashion-MNIST distribuído. Esse estudo permitirá comparar o desempenho de um modelo federado com um modelo centralizado, verificando as vantagens e desvantagens práticas do aprendizado federado em termos de precisão e privacidade.

4.2 Fonte de Dados

A coleta de dados será dividida em duas partes: fontes teóricas e fontes práticas. Para as fontes teóricas, serão consultados artigos científicos, livros, dissertações e teses indexados em bases de dados acadêmicas, como IEEE Xplore, ACM Digital Library, SpringerLink, e Google Scholar. A seleção desses artigos seguirá critérios detalhados mais adiante. Já a fonte de dados prática será o conjunto de dados Fashion-MNIST. A base de dados será aplicada tanto no treinamento centralizado quanto no treinamento federado, permitindo uma comparação direta entre os métodos.

4.3 Critérios de Inclusão e Exclusão

Os critérios para a seleção de artigos científicos na revisão de literatura seguirão a metodologia PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses), com base nos seguintes critérios:

Critérios de Inclusão:

- Artigos que tratam diretamente do aprendizado federado e suas aplicações em IA.
- Estudos que discutem a privacidade de dados em cenários de treinamento distribuído.

- Publicações entre 2015 e 2024, dado o crescimento recente da pesquisa em aprendizado federado.
- Artigos revisados por pares e disponíveis em bases de dados confiáveis (IEEE, ACM, Springer, etc.).
- Estudos que utilizam frameworks práticos como TensorFlow Federated.

Critérios de Exclusão:

- Artigos que não abordam a privacidade de dados ou aprendizado federado.
- Publicações anteriores a 2015, a menos que sejam referências essenciais ou históricas.
- Estudos que não apresentam implementações práticas ou são puramente teóricos sem aplicação clara.

4.4 Procedimentos de Coleta de Dados

A coleta de dados teóricos será realizada em três fases principais: 1. Busca inicial: serão utilizados termos de pesquisa como "federated learning", "data privacy in AI", "decentralized machine learning", e "TensorFlow Federated". 2. Análise de relevância: cada artigo será avaliado com base nos títulos e resumos para verificar a aderência aos critérios de inclusão e exclusão. 3. Leitura aprofundada: os artigos selecionados serão lidos na íntegra, com ênfase nos objetivos, metodologia e resultados, focando nas discussões que envolvem aprendizado federado, privacidade e comparações entre modelos centralizados e federados.

Para a parte prática, o conjunto de dados Fashion-MNIST será particionado de forma a simular um cenário de dados descentralizados, onde os dados serão distribuídos em vários "clientes". Esses clientes terão acesso a subconjuntos dos dados completos, conforme a configuração típica de um sistema de aprendizado federado.

4.5 Análise de Dados

Os dados coletados serão analisados de duas formas principais:

1. Revisão de literatura: será realizada uma análise crítica dos artigos selecionados, categorizando-os de acordo com as abordagens de privacidade, frameworks utilizados, resultados de desempenho e propostas para melhorias no aprendizado federado. Será feita uma comparação entre os diferentes métodos de preservação de privacidade, identificando lacunas de pesquisa.

2. Estudo experimental: os dados práticos serão analisados com base nos resultados de desempenho dos modelos treinados. As métricas de avaliação incluirão acurácia, precisão, sensibilidade, F1-score e tempo de treinamento. Ferramentas como TensorFlow e TensorFlow Federated serão utilizadas para construir os modelos e calcular essas métricas. A análise comparativa entre o modelo centralizado e o federado será realizada para verificar as vantagens e desvantagens de cada abordagem, com foco em privacidade e desempenho.

Esta metodologia permite uma análise abrangente do tema, garantindo que o estudo aborde tanto a teoria quanto a prática do aprendizado federado, enquanto investiga os desafios de manter a privacidade dos dados em cenários descentralizados.

5 Desenvolvimento dos modelos de treinamento

O problema central que o modelo de aprendizado federado busca resolver reside na exposição potencial de dados sensíveis quando centralizados em um único servidor para treinamento de modelos de IA. Em vez de enviar os dados dos dispositivos para um servidor central, onde poderiam ser acessados ou comprometidos, o aprendizado federado propõe um paradigma descentralizado no qual os dispositivos realizam o treinamento localmente, compartilhando apenas os parâmetros do modelo (como pesos e gradientes) com o servidor central. Esta abordagem permite que os dispositivos colaborem no desenvolvimento de um modelo global sem nunca expor os dados locais.

A estruturação do problema foi conduzida considerando-se um cenário prático, onde múltiplos dispositivos, cada um contendo dados não identicamente distribuídos (não-IID), contribuem para o treinamento de um modelo de classificação de imagens (utilizando o conjunto de dados Fashion-MNIST como referência). Este cenário foi escolhido por refletir uma situação comum em aplicações reais, onde dados como registros de saúde, preferências de usuários ou informações financeiras são distribuídos entre dispositivos pessoais e não podem ser compartilhados diretamente por questões de privacidade e conformidade com regulamentos como o GDPR. Assim, o método desenvolvido abrange a implementação de um modelo federado que deve ser capaz de aprender de maneira eficiente e precisa a partir de dados dispersos, preservando a privacidade, e enfrenta desafios como a heterogeneidade dos dispositivos, a variabilidade da conectividade de rede e a necessidade de garantir a convergência do modelo global. A escolha do aprendizado federado, mais especificamente do processo Federated Averaging (FedAvg), é justificada pela sua capacidade de operar eficientemente em ambientes com essas características, buscando balancear a necessidade de proteção dos dados com a performance do modelo treinado.

5.1 Implementação do Modelo de Aprendizado Federado

Para melhor entendimento, foi dividido em seções a implementação do código do modelo de aprendizado federado. Cada seção abordará uma etapa do desenvolvimento do modelo e por fim será apresentado o código completo.

5.1.1 Definição do Modelo Keras

O primeiro passo na implementação do modelo federado foi a criação de um modelo Keras básico, que servirá como base para o treinamento federado. O modelo foi projetado

para a base de dados Fashion-MNIST, que contém imagens de roupas divididas em 10 classes. A estrutura escolhida para esse modelo é uma Convolutional Neural Network (CNN), que é ideal para processar imagens. A CNN foi configurada com duas camadas de convolução seguidas por camadas de pooling, um padrão comum em tarefas de classificação de imagens, pois essas camadas são muito eficazes na extração de características visuais. A última parte do modelo é uma camada densa com 128 neurônios e a camada de saída com 10 neurônios, correspondendo às classes da base de dados.

Listing 5.1 – Função para criar um modelo Keras

```
def create_keras_model():
    model = tf.keras.models.Sequential([
        tf.keras.layers.Input(shape=(28, 28, 1)),
        tf.keras.layers.Conv2D(32, kernel_size=(3, 3),
            activation='relu'),
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
        tf.keras.layers.Conv2D(64, kernel_size=(3, 3),
            activation='relu'),
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dense(10, activation='softmax')
    ])
    return model
```

A escolha de uma CNN foi feita devido ao seu ótimo desempenho em tarefas de visão computacional, como é o caso do Fashion-MNIST, onde características como bordas, texturas e padrões podem ser capturados e processados de forma eficiente pelas camadas convolucionais.

5.1.2 Construção do Modelo Federado

Uma escolha importante ao implementar um modelo federado é como serão avaliadas as métricas de desempenho. Além da acurácia tradicional (*Sparse Categorical Accuracy*), foi adicionada a métrica de Acurácia *Top-3* (*Sparse Top-K Categorical Accuracy*), que avalia se a classe correta está entre as três previsões mais prováveis feitas pelo modelo. Essa escolha é interessante em contextos onde o erro de classificação é aceitável se a previsão correta estiver próxima do topo, o que pode ser útil em sistemas de recomendação ou em ambientes onde há muitas classes e as previsões precisam ser refinadas ao longo do tempo.

Listing 5.2 – Construção do modelo federado

```
def create_federated_model():
keras_model = create_keras_model()

return tff.learning.models.from_keras_model(
    keras_model,
    input_spec=(tf.TensorSpec(shape=[None, 28, 28, 1],
                              dtype=tf.float32),
                tf.TensorSpec(shape=[None], dtype=tf.int64)),
    loss=tf.keras.losses.SparseCategoricalCrossentropy(),
    metrics=[
        tf.keras.metrics.SparseCategoricalAccuracy(),
        tf.keras.metrics.SparseTopKCategoricalAccuracy(k=3)
    ]
)
```

Ao incluir essa métrica no modelo federado, a expectativa era observar uma melhora no desempenho geral ao lidar com a base de dados distribuída, onde as variabilidades dos dados entre os dispositivos podem dificultar a obtenção de acurácia elevada de imediato. A Acurácia Top-3 oferece uma margem para avaliar se o modelo está capturando informações corretas em um cenário mais flexível.

5.1.3 Construção do Processo de Federado

O processo federado foi implementado utilizando o algoritmo Federated Averaging, que combina os pesos atualizados de cada cliente em um servidor central, onde os modelos locais são treinados nos dados locais e suas atualizações são agregadas. A escolha do Federated Averaging se deve à sua eficiência e simplicidade, especialmente ao lidar com grandes quantidades de dados distribuídos. Além disso, ele permite que os dispositivos participantes treinem seus modelos localmente e compartilhem apenas os gradientes ou atualizações dos pesos, preservando assim a privacidade dos dados dos usuários.

Listing 5.3 – Construção do processo federado

```
def build_federated_averaging_process():
    return tff.learning.algorithms.build_unweighted_fed_avg(
        model_fn=create_federated_model,
        client_optimizer_fn=lambda:
            tf.keras.optimizers.Adam(learning_rate=0.001),
```

```
        server_optimizer_fn=lambda:
            tf.keras.optimizers.SGD(learning_rate=1.0)
)
```

Foi utilizado o otimizador Adam nos dispositivos locais, uma escolha feita para lidar melhor com as variações nos dados entre diferentes dispositivos, já que o Adam tem uma capacidade adaptativa de ajustar a taxa de aprendizado com base nas características dos dados. O otimizador SGD foi usado no servidor para combinar as atualizações dos modelos, garantindo uma convergência mais estável.

5.1.4 Preparação dos Dados

A etapa de preparação dos dados é crítica em um modelo federado. Aqui, os dados foram normalizados e divididos entre 10 clientes, simulando diferentes dispositivos locais. Essa divisão é importante para simular um ambiente federado realista, onde diferentes dispositivos possuem subconjuntos dos dados totais e, portanto, variabilidades nas distribuições de dados.

Listing 5.4 – Preparação dos dados

```
def preprocess(dataset):
    return dataset.map(lambda x, y:
        (tf.cast(tf.expand_dims(x, -1), tf.float32),
         tf.cast(y, tf.int64))).batch(20)

def create_client_data(images, labels, num_clients=10,
        num_samples_per_client=6000):
    client_data = []
    for i in range(num_clients):
        client_images =
            images[i*num_samples_per_client:(i+1)*
                   num_samples_per_client]
        client_labels =
            labels[i*num_samples_per_client:(i+1)*
                   num_samples_per_client]
        dataset =
            tf.data.Dataset.from_tensor_slices(
                (client_images, client_labels))

        client_data.append(preprocess(dataset))
```

```
    return client_data
```

A escolha da base de dados Fashion-MNIST foi feita pela simplicidade e familiaridade desse conjunto no treinamento de modelos de classificação de imagens. Ele oferece um bom balanço entre complexidade visual e tamanho, sendo leve o suficiente para ser usado em dispositivos locais sem grandes recursos computacionais.

5.1.5 Treinamento do Modelo Federado

Listing 5.5 – Treinamento do modelo federado

```
iterative_process = build_federated_averaging_process()
state = iterative_process.initialize()

for round_num in range(1, 21):
    state, metrics =
        iterative_process.next(state, federated_train_data)
    print(f'Rodada {round_num}: {metrics}')
```

O treinamento é executado através de um processo iterativo, onde o modelo é treinado em múltiplas rodadas. A escolha de 20 rodadas de treinamento permite uma observação inicial do comportamento do modelo federado sem sobrecarregar o sistema. Em cada rodada, os clientes locais treinam o modelo com seus dados privados e enviam as atualizações para o servidor central, onde as atualizações são agregadas para formar um modelo global atualizado. O uso da função `next` no TensorFlow Federated aplica a agregação de *FedAvg*, onde as médias das atualizações dos clientes são calculadas. O feedback das métricas de cada rodada permite acompanhar a evolução do modelo, especialmente em termos de precisão (via *SparseCategoricalAccuracy*), oferecendo uma visão clara de como o modelo está se ajustando aos dados federados ao longo do tempo.

5.2 Implementação do Modelo de Aprendizado Centralizado

Para criar um modelo de treinamento centralizado, foi realizado um processo semelhante ao do aprendizado federado utilizando a mesma biblioteca de dados (MNIST) e as mesmas ferramentas, apenas sem a distribuição de dados entre dispositivos. Esse modelo será utilizado para comparação com o modelo federado desenvolvido anteriormente, possibilitando a avaliação do impacto do aprendizado centralizado versus o aprendizado federado na preservação da privacidade e desempenho.

5.2.1 Criação do Modelo de Aprendizado Centralizado - CNN

Neste projeto foram implementados 3 modelos de treinamento centralizado, sendo que cada um deles utiliza uma técnica de treinamento sendo elas: uma Rede Neural Convolucional (CNN), uma Rede Neural Recorrente (RNN) e um modelo Rede neural multicamadas (MLP). A seguir, será apresentado o código referente à implementação do modelo de CNN centralizado afim de ilustrar como funciona o treinamento centralizado. Ao final do capítulo, será apresentado o código completo dos 3 modelos com uma breve explicação de cada um.

5.2.1.1 Definição do Modelo Keras

A arquitetura da rede convolucional é simples, composta por duas camadas convolucionais seguidas de camadas de pooling, o que é uma escolha típica para reconhecimento de imagens. A primeira camada convolucional usa 32 filtros, e a segunda utiliza 64 filtros, mantendo o modelo relativamente leve. Isso é importante considerando o uso de recursos limitados no Google Colab, onde o tempo de treinamento e a capacidade de processamento são fatores críticos. O uso de camadas de pooling reduz a dimensionalidade dos dados, diminuindo o número de parâmetros e o tempo de computação necessário.

Listing 5.6 – Definição do modelo Keras centralizado

```
def create_cnn_model():
    return tf.keras.models.Sequential([
        tf.keras.layers.Input(shape=(28, 28, 1)),
        tf.keras.layers.Conv2D(32, kernel_size=(3, 3),
            activation='relu'),
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
        tf.keras.layers.Conv2D(64, kernel_size=(3, 3),
            activation='relu'),
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(10, activation='softmax')
    ])
```

O uso de dropout com uma taxa de 0.5 na camada totalmente conectada foi implementado para evitar overfitting, uma vez que o dataset Fashion-MNIST é relativamente pequeno, e sem técnicas de regularização, o modelo poderia memorizar os dados de treino, reduzindo sua capacidade de generalizar. O dropout desativa aleatoriamente 50%

dos neurônios durante o treinamento, o que força o modelo a aprender representações mais robustas dos dados, melhorando seu desempenho nos dados de validação.

5.2.1.2 Criação e Compilação do Modelo Centralizado

A função de perda escolhida foi a entropia cruzada esparsa (sparse_categorical_crossentropy), adequada para problemas de classificação multiclasse. Ela mede a diferença entre as previsões do modelo e as classes verdadeiras, penalizando mais fortemente previsões incorretas para classes que têm maior probabilidade atribuída. O otimizador Adam foi escolhido por ser um método de otimização robusto que combina as vantagens do RMSProp e do SGD com momentum, ajustando a taxa de aprendizado dinamicamente durante o treinamento. A taxa de aprendizado inicial foi ajustada para 0.001, que é um valor conservador e permite que o modelo aprenda de forma mais gradual e estável.

Listing 5.7 – Criação e compilação do modelo centralizado

```
model_cnn = create_cnn_model()
model_cnn.compile(optimizer=
                  tf.keras.optimizers.Adam(learning_rate=0.001),
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
```

5.2.1.3 Treinamento do Modelo Centralizado

Durante o treinamento, a precisão foi monitorada tanto nos dados de treinamento quanto nos de validação. A escolha pela métrica de acurácia faz sentido, dado que o problema é de classificação e a acurácia mede diretamente a proporção de previsões corretas. Além disso, a função de perda também foi monitorada para garantir que o modelo não só esteja melhorando sua acurácia, mas também minimizando a entropia cruzada ao longo das épocas.

Listing 5.8 – Treinamento do modelo centralizado

```
history_cnn =
    model_cnn.fit(fashion_train,
                  fashion_train_labels,
                  epochs=50,
                  batch_size=32,
                  validation_data=
                  (fashion_test, fashion_test_labels))
```

No modelo federado, o treinamento é distribuído entre vários clientes e o processo de treinamento é coordenado centralmente. A atualização dos pesos é feita de maneira agregada, o que é uma abordagem fundamentalmente diferente da atualização direta utilizada no treinamento centralizado.

5.2.1.4 Avaliação do Modelo

Listing 5.9 – Avaliação do modelo centralizado

```
test_loss_cnn, test_acc_cnn =
    model_cnn.evaluate(mnist_test, mnist_test_labels)
```

A avaliação do modelo é feita usando `model.evaluate()`, que calcula a perda e a precisão do modelo nos dados de teste. Isso fornece uma medida direta do desempenho do modelo em dados que não foram usados durante o treinamento, ajudando a verificar sua capacidade de generalização. No contexto do aprendizado federado, a avaliação é geralmente realizada de forma agregada após o treinamento federado ter sido concluído. A abordagem é diferente, pois a avaliação pode ser feita em vários conjuntos de dados de clientes para refletir a performance geral do modelo distribuído.

5.3 Justificativa

A escolha do processo de **Federated Averaging (FedAvg)** foi baseada em sua eficácia e simplicidade para a implementação de aprendizado federado em cenários onde a privacidade dos dados é uma preocupação central. A seguir, discutiremos as razões específicas para a escolha do FedAvg, suas vantagens e desvantagens, e os modelos de treinamento tradicionais que poderiam ser utilizados para comparar a performance.

5.3.1 Razões para a Escolha do FedAvg

O **FedAvg** é amplamente reconhecido e utilizado como uma das abordagens mais eficientes para o aprendizado federado, especialmente em contextos onde os dados são distribuídos entre vários clientes (dispositivos) que possuem capacidade computacional limitada. Ele opera através da média ponderada dos pesos dos modelos treinados localmente em cada cliente. Algumas das principais razões para a escolha do FedAvg incluem:

1. Simplicidade de Implementação: O FedAvg é relativamente fácil de implementar, tanto em termos de código quanto em infraestrutura. Ele não requer modificações complexas no modelo original e é compatível com a maioria dos modelos de aprendizado de máquina.

2. Eficiência em Ambientes Heterogêneos: Em cenários onde os clientes possuem capacidades computacionais variadas e dados não IID (independentemente e identicamente distribuídos), o FedAvg tem se mostrado robusto e eficaz. Ele permite que cada cliente contribua de acordo com sua capacidade, o que é ideal para dispositivos com diferentes níveis de poder computacional e conectividade.

3. Privacidade Preservada: Como o FedAvg realiza o treinamento localmente nos dispositivos dos clientes e apenas os pesos dos modelos (não os dados) são enviados para o servidor central, ele ajuda a preservar a privacidade dos dados.

5.3.2 Vantagens do FedAvg

1. Escalabilidade: FedAvg é altamente escalável, sendo capaz de lidar com milhares de dispositivos clientes. A abordagem distribuída minimiza a necessidade de centralização de dados, reduzindo os gargalos associados ao processamento em larga escala.

2. Flexibilidade: FedAvg pode ser aplicado a uma ampla variedade de modelos de aprendizado de máquina, desde redes neurais profundas até modelos mais simples, tornando-o versátil em diferentes cenários de aplicação.

3. Resiliência a Dados Não IID: Em muitos cenários do mundo real, os dados disponíveis em diferentes dispositivos não seguem uma distribuição IID. O FedAvg consegue lidar razoavelmente bem com essas discrepâncias.

5.3.3 Desvantagens do FedAvg

1. Convergência Mais Lenta: Devido à natureza distribuída e à variabilidade entre os dispositivos, a convergência do FedAvg pode ser mais lenta comparada a métodos centralizados, especialmente em casos onde os dados são altamente desbalanceados entre os clientes.

2. Dependência de Conectividade: O FedAvg requer que os dispositivos clientes enviem periodicamente seus modelos atualizados ao servidor central. Em ambientes com conectividade de rede instável ou dispositivos com pouca energia, isso pode se tornar um desafio.

3. Sobrecarga Computacional e de Comunicação: Embora o FedAvg minimize a necessidade de centralização de dados, ele ainda requer que os dispositivos locais realizem computações significativas e participem regularmente da comunicação com o servidor, o que pode ser oneroso para dispositivos de baixa potência.

5.4 Código e aplicação dos modelos

Esta seção apresenta o código completo desenvolvido para a implementação do modelo de aprendizado federado utilizando o TensorFlow Federated.

A segunda parte dessa seção apresenta a implementação do modelo de treinamento centralizado que será utilizado para comparação com o modelo federado. O código é estruturado de forma semelhante ao modelo federado, mas com a diferença de que os dados são centralizados em um único servidor para treinamento.

5.4.1 Implementação completa do Modelo de Aprendizado Federado

Este código implementa um modelo de aprendizado federado utilizando o *TensorFlow Federated* (TFF). O processo começa com a criação de um modelo *Keras* convolucional (CNN) para o dataset *Fashion-MNIST*, que classifica imagens em 10 categorias de roupas e acessórios. A função `create_federated_model` transforma este modelo em uma arquitetura federada, que distribui o treinamento entre vários "clientes", preservando os dados locais. O modelo utiliza a acurácia categórica e a acurácia *Top-3* como métricas, além da perda categórica esparsa.

Em seguida, o código constrói o processo federado utilizando o algoritmo *FedAvg* (média federada), e as funções de otimizador são configuradas tanto para o servidor quanto para os clientes. Os dados do *Fashion-MNIST* são processados e distribuídos entre 10 clientes, simulando diferentes fontes de dados.

Por fim, o código realiza o treinamento federado por 20 rodadas, coletando e armazenando métricas como a acurácia, a acurácia *Top-3*, e a perda. O tempo total de treinamento é medido, e gráficos são gerados para visualizar a evolução das métricas ao longo das rodadas de treinamento, fornecendo insights sobre o desempenho do modelo federado.

```
import tensorflow as tf
import tensorflow_federated as tff
import matplotlib.pyplot as plt
import time

def create_keras_model():
    model = tf.keras.models.Sequential([
        tf.keras.layers.Input(shape=(28, 28, 1)),
        tf.keras.layers.Conv2D(32, kernel_size=(3, 3),
            activation='relu'),
```

```
            tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
            tf.keras.layers.Conv2D(64, kernel_size=(3, 3),
                activation='relu'),
            tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
            tf.keras.layers.Flatten(),
            tf.keras.layers.Dense(128, activation='relu'),
            tf.keras.layers.Dense(10, activation='softmax')
        ])
        return model

    def create_federated_model():
        keras_model = create_keras_model()

        return tff.learning.models.from_keras_model(
            keras_model,
            input_spec=
                (tf.TensorSpec(shape=[None, 28, 28, 1],
                    dtype=tf.float32),
                tf.TensorSpec(shape=[None], dtype=tf.int64)),
            loss=tf.keras.losses.SparseCategoricalCrossentropy(),
            metrics=[
                tf.keras.metrics.SparseCategoricalAccuracy(),
                tf.keras.metrics.SparseTopKCategoricalAccuracy(k=3)
            ]
        )

    def build_federated_averaging_process():
        return tff.learning.algorithms.build_unweighted_fed_avg(
            model_fn=create_federated_model,
            client_optimizer_fn=lambda:
                tf.keras.optimizers.Adam(learning_rate=0.001),
            server_optimizer_fn=lambda:
                tf.keras.optimizers.SGD(learning_rate=1.0)
        )

    def preprocess(dataset):
        return dataset.map(lambda x, y:
            (tf.cast(tf.expand_dims(x, -1), tf.float32),
                tf.cast(y, tf.int64))).batch(20)
```

```
def create_client_data(images, labels,
        num_clients=10, num_samples_per_client=6000):
    client_data = []
    for i in range(num_clients):
        client_images =
            images[i*num_samples_per_client:(i+1)*
                num_samples_per_client]
        client_labels =
            labels[i*num_samples_per_client:(i+1)*
                num_samples_per_client]
        dataset =
            tf.data.Dataset.from_tensor_slices((client_images,
                client_labels))
        client_data.append(preprocess(dataset))
    return client_data

(fashion_train_images, fashion_train_labels),
(fashion_test_images, fashion_test_labels) =
    tf.keras.datasets.fashion_mnist.load_data()

fashion_train_images = fashion_train_images / 255.0
fashion_test_images = fashion_test_images / 255.0

federated_train_data =
    create_client_data(fashion_train_images, fashion_train_labels)
federated_test_data =
    preprocess(tf.data.Dataset.from_tensor_slices(
            (fashion_test_images, fashion_test_labels)))

iterative_process = build_federated_averaging_process()

state = iterative_process.initialize()

accuracy_per_round = []
top_k_accuracy_per_round = []
loss_per_round = []

start_time = time.time()
```

```
print('Rodando treinamento federado...')
for round_num in range(1, 21):
    state, metrics =
        iterative_process.next(state, federated_train_data)
    print(f'Rodada {round_num}/20: {metrics}')

    accuracy_per_round.append(
        metrics['client_work']
               ['train']
               ['sparse_categorical_accuracy'])
    top_k_accuracy_per_round.append(
        metrics['client_work']
               ['train']
               ['sparse_top_k_categorical_accuracy'])
    loss_per_round.append(
        metrics['client_work']
               ['train']
               ['loss'])

end_time = time.time()
total_training_time = end_time - start_time
print(f'Tempo total de treinamento:
    {total_training_time:.2f} segundos')

def plot_metrics(metric_values, metric_name, title, ylabel, color):
    plt.figure(figsize=(10, 6))
    plt.plot(range(1, len(metric_values) + 1),
             metric_values,
             marker='o',
             linestyle='-',
             color=color,
             label=metric_name)
    plt.title(title)
    plt.xlabel('Rodada')
    plt.ylabel(ylabel)
    plt.grid(True)
    plt.legend()
    plt.show()
```

```
plot_metrics(accuracy_per_round, 'Acuracia',
    'Acuracia por Rodada de Treinamento Federado',
    'Acuracia', 'b')
plot_metrics(top_k_accuracy_per_round, 'Acuracia Top-3',
    'Acuracia Top-3 por Rodada de Treinamento Federado',
    'Acuracia Top-3', 'g')
plot_metrics(loss_per_round, 'Perda',
    'Perda por Rodada de Treinamento Federado',
    'Perda', 'r')
```

5.4.2 Implementação completa do Modelo de Aprendizado Centralizado - CNN

O código implementa um modelo de rede neural convolucional (CNN) utilizando o *TensorFlow* para o dataset *Fashion-MNIST*, que contém imagens de roupas classificadas em 10 categorias. O processo começa carregando e normalizando os dados para valores entre 0 e 1. Em seguida, o código adiciona uma dimensão aos dados, necessária para que a CNN processe as imagens.

O modelo CNN é definido com várias camadas, incluindo duas camadas convolucionais com 32 e 64 filtros, respectivamente, cada uma seguida por uma camada de *max pooling* para redução dimensional. Após as camadas convolucionais, há uma camada totalmente conectada (*dense*) com 128 neurônios e uma função de ativação *ReLU*. O *dropout* é utilizado para prevenir *overfitting*, e a última camada utiliza *softmax* para classificar as 10 categorias.

O modelo é compilado usando o otimizador *Adam* e a função de perda de *entropia cruzada categórica esparsa*. O treinamento é realizado ao longo de 50 épocas com os dados de treinamento, e o modelo é avaliado nos dados de teste, fornecendo as métricas de acurácia e perda. Ao final, são plotados gráficos para visualizar a evolução da acurácia e da perda durante o treinamento e a validação.

Essas escolhas de arquitetura e parâmetros visam otimizar a capacidade de generalização do modelo sem *overfitting*, ao mesmo tempo em que o modelo é suficientemente robusto para o conjunto de dados relativamente simples como o *Fashion-MNIST*.

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import time
```

```
(fashion_train, fashion_train_labels),
(fashion_test, fashion_test_labels) =
    tf.keras.datasets.fashion_mnist.load_data()

fashion_train = fashion_train / 255.0
fashion_test = fashion_test / 255.0

fashion_train = fashion_train[..., np.newaxis]
fashion_test = fashion_test[..., np.newaxis]

def create_cnn_model():
    return tf.keras.models.Sequential([
        tf.keras.layers.Input(shape=(28, 28, 1)),
        tf.keras.layers.Conv2D(32, kernel_size=(3, 3),
            activation='relu'),
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
        tf.keras.layers.Conv2D(64, kernel_size=(3, 3),
            activation='relu'),
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(10, activation='softmax')
    ])

model_cnn = create_cnn_model()

model_cnn.compile(optimizer=
        tf.keras.optimizers.Adam(learning_rate=0.001),
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy'])

start_time = time.time()

history_cnn = model_cnn.fit(fashion_train,
                fashion_train_labels,
                epochs=50,
                batch_size=32,
```

```
                validation_data=
                    (fashion_test, fashion_test_labels))

end_time = time.time()
total_time_cnn = end_time - start_time
print(f'\nTempo total de treinamento (CNN):
    {total_time_cnn:.2f} segundos')

test_loss_cnn, test_acc_cnn =
    model_cnn.evaluate(fashion_test, fashion_test_labels)
print(f'\nPrecisao do modelo CNN
    nos dados de teste: {test_acc_cnn}')

def plot_metrics(history, metric_name, title, ylabel):
    plt.figure(figsize=(10, 5))
    plt.plot(history.history[metric_name],
        label=f'{ylabel} de Treinamento (CNN)')
    plt.plot(history.history[f'val_{metric_name}'],
        label=f'{ylabel} de Validacao (CNN)')
    plt.title(title)
    plt.xlabel('Epocas')
    plt.ylabel(ylabel)
    plt.xticks(np.arange(0,
        len(history.history[metric_name]), step=1))
    plt.legend()
    plt.grid(True)
    plt.show()

plot_metrics(history_cnn, 'accuracy',
    'Acuracia durante o Treinamento e Validacao (CNN)',
    'Acuracia')
plot_metrics(history_cnn, 'loss',
    'Perda durante o Treinamento e Validacao (CNN)',
    'Perda')
```

5.4.3 Implementação completa do Modelo de Aprendizado Centralizado - MLP

Este código implementa um modelo de *Perceptron Multicamadas* (MLP) utilizando a biblioteca *TensorFlow* para o dataset *Fashion-MNIST*. O código é estruturado em diversas etapas, começando pelo carregamento e normalização dos dados, que são dimensionados para o intervalo de 0 a 1, como forma de otimizar o processo de treinamento.

Após isso, os dados de entrada são remodelados em uma estrutura linear adequada para um MLP (removendo a dimensão do canal de cores e transformando cada imagem em um vetor unidimensional de 784 valores). Em seguida, o modelo MLP é definido com quatro camadas densas, sendo as três primeiras camadas com 512, 256 e 128 neurônios e ativação *ReLU*, para introduzir não-linearidade, e uma última camada de 10 neurônios, utilizando *softmax* para a classificação das imagens em uma das 10 categorias do *Fashion-MNIST*. O *dropout* é utilizado para regularização e prevenir *overfitting*.

O modelo é compilado com o otimizador *Adam* e a função de perda de *entropia cruzada categórica esparsa*, que é ideal para problemas de classificação com múltiplas classes. O treinamento é realizado por 50 épocas, e as métricas de desempenho, como *acurácia* e *perda*, são monitoradas tanto nos dados de treinamento quanto de validação. Por fim, gráficos são gerados para visualizar a evolução da acurácia e da perda ao longo das épocas, oferecendo uma visão clara do comportamento do modelo durante o treinamento e validação.

Esse modelo MLP é uma alternativa mais simples em relação às redes convolucionais, focando em capturar padrões globais dos dados sem levar em conta a estrutura espacial das imagens, como fazem as CNNs.

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import time

(fashion_train, fashion_train_labels),
(fashion_test, fashion_test_labels) =
    tf.keras.datasets.fashion_mnist.load_data()

fashion_train = fashion_train / 255.0
fashion_test = fashion_test / 255.0

fashion_train_mlp =
```

```
    fashion_train.reshape(-1, 28 * 28)
fashion_test_mlp =
    fashion_test.reshape(-1, 28 * 28)

def create_mlp_model():
    return tf.keras.models.Sequential([
        tf.keras.layers.Input(shape=(28 * 28,)),
        tf.keras.layers.Dense(512, activation='relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(256, activation='relu'),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dense(10, activation='softmax')
    ])

model_mlp = create_mlp_model()

model_mlp.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

start_time = time.time()

history_mlp = model_mlp.fit(fashion_train_mlp,
                            fashion_train_labels,
                            epochs=50,
                            batch_size=32,
                            validation_data=(fashion_test_mlp,
                                fashion_test_labels))

end_time = time.time()
total_time_mlp = end_time - start_time
print(f'\nTempo total de treinamento (MLP):
    {total_time_mlp:.2f} segundos')

test_loss_mlp, test_acc_mlp =
    model_mlp.evaluate(fashion_test_mlp, fashion_test_labels)
print(f'\nPrecisao do modelo MLP
        nos dados de teste: {test_acc_mlp}')
```

```
def plot_metrics(history, metric_name, title, ylabel):
    plt.figure(figsize=(10, 5))
    plt.plot(history.history[metric_name],
        label=f'{ylabel} de Treinamento (MLP)')
    plt.plot(history.history[f'val_{metric_name}'],
        label=f'{ylabel} de Validacao (MLP)')
    plt.title(title)
    plt.xlabel('Epocas')
    plt.ylabel(ylabel)
    plt.xticks(np.arange(1,
        len(history.history[metric_name])+1, step=1))
    plt.legend()
    plt.grid(True)
    plt.show()

plot_metrics(history_mlp, 'accuracy',
    'Acuracia durante o Treinamento e Validacao (MLP)',
    'Acuracia')
plot_metrics(history_mlp, 'loss',
    'Perda durante o Treinamento e Validacao (MLP)',
    'Perda')
```

5.4.4 Implementação completa do Modelo de Aprendizado Centralizado - DNN

Este código implementa um modelo de MLP profunda, simulando uma rede neural profunda (DNN) para classificar imagens do conjunto de dados *Fashion-MNIST* utilizando o *TensorFlow*. A base de dados é carregada e normalizada, com os valores dos pixels sendo ajustados entre 0 e 1 para melhorar o desempenho durante o treinamento. Diferente de uma CNN (rede neural convolucional), as imagens são diretamente achatadas em vetores unidimensionais com 784 pixels (28x28) antes de serem processadas pelas camadas densas da DNN.

O modelo implementado possui uma arquitetura com quatro camadas densas completamente conectadas (*fully connected*). A primeira camada contém 1024 neurônios com ativação *ReLU*, seguida de uma camada *Dropout* para evitar o *overfitting*. Em seguida, temos camadas de 512, 256, e 128 neurônios com ativação *ReLU*, e a camada final contém 10 neurônios com ativação *softmax* para a saída de classificação em 10 classes, correspondentes às diferentes categorias do *Fashion-MNIST*.

O modelo é treinado usando o otimizador *Adam* e a função de perda *sparse_categorical_crosse*

apropriada para problemas de classificação com várias classes. O treinamento ocorre ao longo de 50 épocas, com monitoramento das métricas de *acurácia* e *perda* tanto no conjunto de treinamento quanto de validação, permitindo verificar a evolução e o ajuste do modelo. Ao final, as métricas são plotadas para oferecer uma visualização clara do desempenho do modelo ao longo do treinamento.

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import time

(fashion_train, fashion_train_labels),
(fashion_test, fashion_test_labels) =
    tf.keras.datasets.fashion_mnist.load_data()

fashion_train = fashion_train / 255.0
fashion_test = fashion_test / 255.0

fashion_train = fashion_train[..., np.newaxis]
fashion_test = fashion_test[..., np.newaxis]

def create_dnn_model():
    return tf.keras.models.Sequential([
        tf.keras.layers.Input(shape=(28, 28, 1)),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(1024, activation='relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(512, activation='relu'),
        tf.keras.layers.Dense(256, activation='relu'),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dense(10, activation='softmax')
    ])

model_dnn = create_dnn_model()

model_dnn.compile(optimizer=
                  tf.keras.optimizers.Adam(learning_rate=0.001),
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
```

```
start_time = time.time()

history_dnn = model_dnn.fit(fashion_train,
                            fashion_train_labels,
                            epochs=50,
                            batch_size=32,
                            validation_data=(fashion_test,
                                fashion_test_labels))

end_time = time.time()
total_time_dnn = end_time - start_time
print(f'\nTempo total de treinamento (DNN):
    {total_time_dnn:.2f} segundos')

test_loss_dnn, test_acc_dnn =
    model_dnn.evaluate(fashion_test, fashion_test_labels)
print(f'\nPrecisao do modelo DNN
    nos dados de teste: {test_acc_dnn}')

def plot_metrics(history, metric_name, title, ylabel):
    plt.figure(figsize=(10, 5))
    plt.plot(history.history[metric_name],
        label=f'{ylabel} de Treinamento (DNN)')
    plt.plot(history.history[f'val_{metric_name}'],
        label=f'{ylabel} de Validacao (DNN)')
    plt.title(title)
    plt.xlabel('Epocas')
    plt.ylabel(ylabel)
    plt.xticks(np.arange(1,
        len(history.history[metric_name]) + 1, step=1))
    plt.legend()
    plt.grid(True)
    plt.show()

plot_metrics(history_dnn, 'accuracy',
    'Acuracia durante o Treinamento e Validacao (DNN)',
    'Acuracia')
plot_metrics(history_dnn, 'loss',
```

```
'Perda durante o Treinamento e Validacao (DNN)',
'Perda')
```

6 Análise dos Resultados e Discussões

Este capítulo apresenta a análise dos resultados obtidos com a execução dos modelos de treinamento centralizado e distribuído. Além disso, é exposta a análise do autor referente aos dados coletados e as impressões obtidas durante a execução dos experimentos. Inicialmente há a apresentação dos resultados obtidos, acompanhada pela Posteriormente, são discutidas as dificuldades e limitações encontradas durante a execução dos experimentos.

Serão apresentados e discutidos os desempenhos dos modelos desenvolvidos ao longo do projeto, tanto no contexto centralizado quanto no federado. Três modelos centralizados foram implementados: uma rede neural convolucional (CNN), uma rede neural multicamadas (MLP) e uma rede neural profunda (DNN). Esses modelos serão comparados com o modelo de aprendizado federado, permitindo uma análise detalhada das métricas obtidas, como acurácia, precisão, sensibilidade, F1-score e o tempo de processamento. A comparação entre os modelos visa identificar as vantagens e limitações de cada abordagem, destacando o impacto do aprendizado federado na preservação de privacidade dos dados e na eficiência do treinamento distribuído.

6.1 Análise dos Resultados

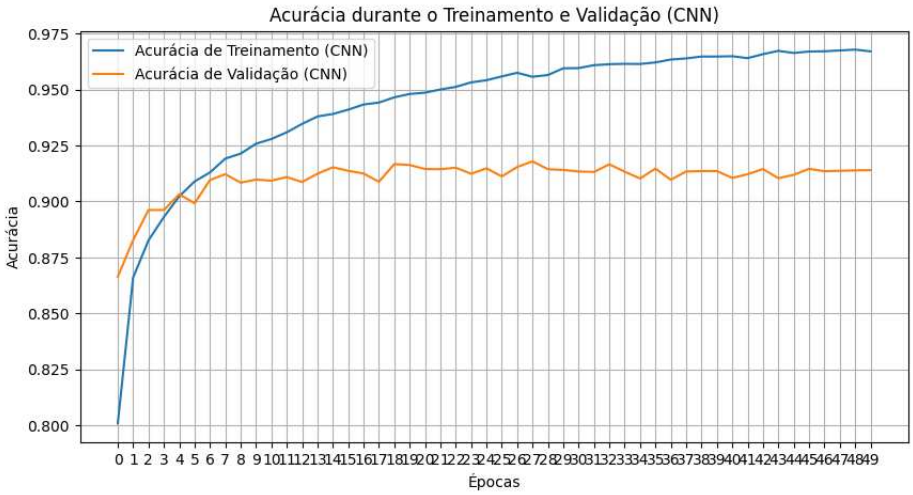
6.2 Desempenho dos Modelos Centralizados

6.2.1 Modelo Centralizado - Rede neural convolucional

Um dos modelos centralizados foi implementado utilizando uma rede neural convolucional (CNN) e treinado sobre o conjunto de dados Fashion-MNIST. O modelo foi desenhado com duas camadas convolucionais seguidas de camadas de pooling, aplicando a técnica de dropout para evitar o overfitting. O modelo foi treinado com 50 épocas, utilizando a otimização Adam, com uma taxa de aprendizado ajustada para 0,001 para garantir um aprendizado gradual. O objetivo foi avaliar o desempenho de um modelo CNN simples, mas com regularização, em um conjunto de dados com complexidade visual moderada, como o Fashion-MNIST.

Durante o treinamento, como observado na Figura 5, o modelo apresentou uma curva de aprendizado consistente, com acurácia de treino subindo de 72,57% na primeira época para cerca de 96,71% na última época. Já nos dados de teste, a acurácia se estabilizou em torno de 91,40%. Esses resultados mostram que o modelo CNN consegue generalizar bem para os dados de teste, sem apresentar sinais claros de overfitting, apesar

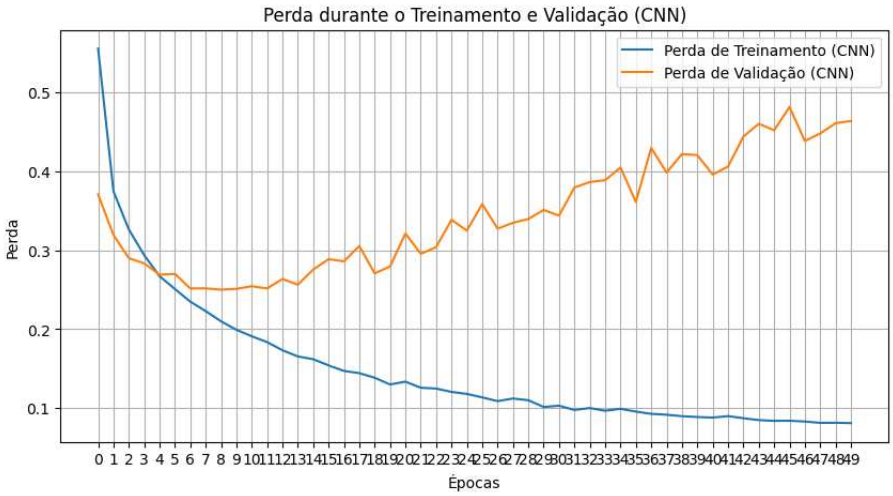
Figura 5 – Curvas de acurácia do modelo centralizado - CNN



Fonte: Elaborado pelo autor.

de ter utilizado dropout para regularização. Isso é evidenciado também pelo comportamento da perda, que foi reduzida de 0,7628 na primeira época para 0,0811 na última.

Figura 6 – Curvas de perda do modelo centralizado - CNN



Fonte: Elaborado pelo autor.

A análise das métricas de validação mostra que o modelo atingiu 91,33% de acurácia nos dados de teste (6). A perda de validação, embora tenha flutuado ao longo das épocas, manteve-se em níveis aceitáveis, finalizando em 0,4632. A validação revelou que o modelo manteve uma consistência de generalização ao longo do treinamento, mas a perda levemente elevada no final das épocas indica que ainda há espaço para melhorias na otimização do modelo, ou ajustes adicionais para regularização mais eficiente.

Em termos de tempo de treinamento, o modelo foi treinado por aproximadamente 3816 segundos (cerca de 63 minutos), o que reflete a complexidade do modelo CNN em

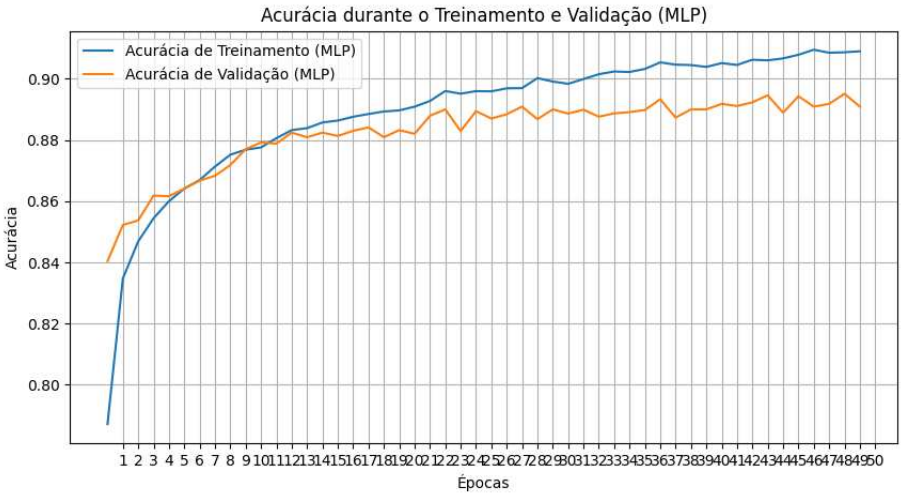
comparação com arquiteturas mais simples como MLP. Apesar desse tempo considerável, os resultados sugerem que o esforço computacional é justificado pelo ganho de desempenho, especialmente na tarefa de classificação de imagens mais complexas, como as da base Fashion-MNIST. Comparando com o modelo federado, a principal vantagem aqui está no tempo de treinamento, uma vez que a centralização dos dados acelera significativamente o processo, eliminando a latência e a complexidade da comunicação entre dispositivos, que são características do aprendizado federado.

6.2.2 Modelo Centralizado - Rede neural multicamadas

O modelo MLP foi construído com uma arquitetura composta por camadas densas totalmente conectadas, com três camadas ocultas contendo 512, 256 e 128 neurônios, respectivamente. Além disso, foi implementado um mecanismo de regularização através de dropout, que visa reduzir o overfitting ao desativar uma fração de neurônios durante o treinamento. A função de ativação utilizada foi a ReLU, com a camada final sendo uma softmax, apropriada para classificação multiclasse.

Durante o treinamento, que ocorreu ao longo de 50 épocas, o modelo mostrou uma melhoria gradual em termos de acurácia, partindo de 73,37% na primeira época e atingindo 91,02% ao final. A precisão nos dados de teste alcançou 89,09%, o que reflete a capacidade do modelo de generalizar razoavelmente bem para dados não vistos (7). A perda de validação oscilou ao longo do processo de treinamento, com uma ligeira tendência de queda, indicando que o modelo estava efetivamente aprendendo, embora a diferença entre perda de treinamento e perda de validação sugira que um ajuste fino adicional poderia melhorar ainda mais o desempenho.

Figura 7 – Curvas de acurácia do modelo centralizado - MLP

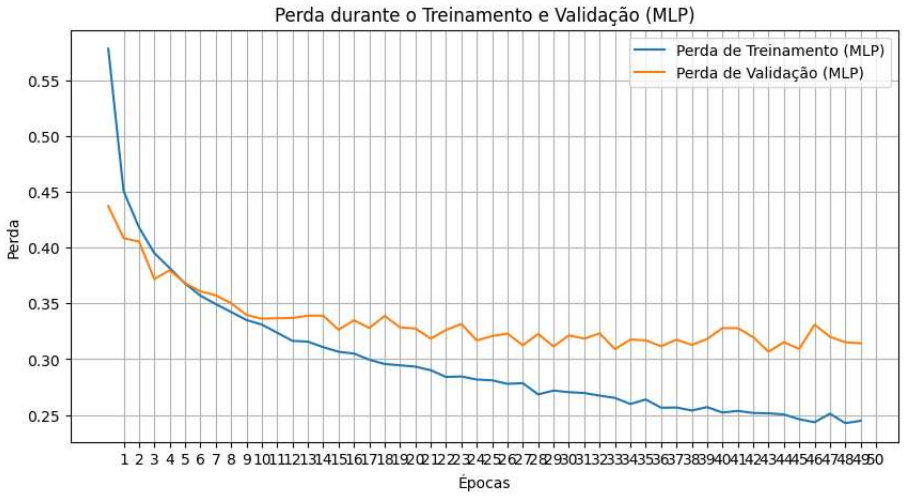


Fonte: Elaborado pelo autor.

Analisando as métricas obtidas, a acurácia se estabilizou em torno de 89% nos

dados de teste, sugerindo que o MLP foi capaz de capturar padrões relevantes para a tarefa de classificação de imagens da base de dados Fashion-MNIST. A regularização por dropout também desempenhou um papel fundamental na mitigação do overfitting, permitindo que o modelo mantivesse um bom desempenho ao lidar com dados de teste. No entanto, em comparação com outros modelos como a CNN, o MLP mostra-se ligeiramente inferior, especialmente em termos de generalização para novas amostras, o que é esperado, dado que a CNN é mais eficaz para capturar padrões espaciais em dados de imagem.

Figura 8 – Curvas de perda do modelo centralizado - MLP



Fonte: Elaborado pelo autor.

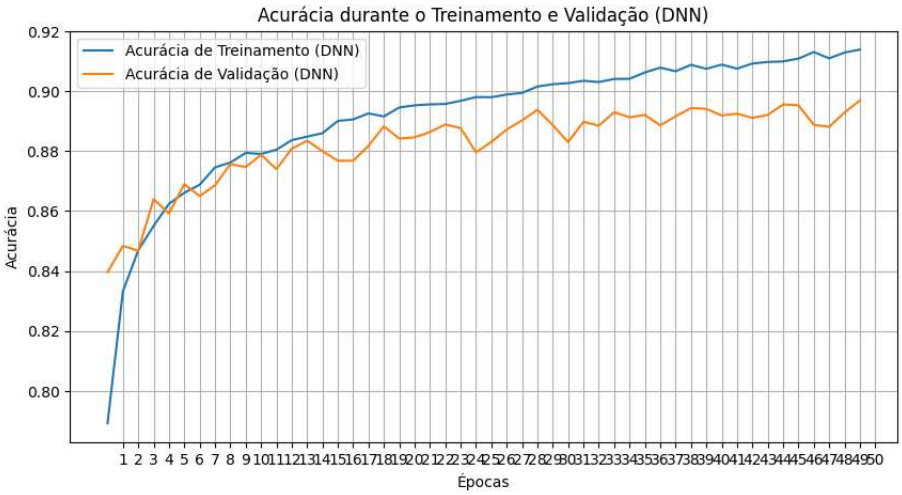
Em termos de tempo de treinamento, o MLP foi significativamente mais rápido do que a CNN, completando o treinamento em aproximadamente 22 minutos (1348 segundos). Embora esse modelo tenha se mostrado eficiente em termos de tempo, a compensação foi uma precisão ligeiramente inferior e uma maior suscetibilidade ao overfitting. Isso reflete uma característica comum dos MLPs em comparação com arquiteturas mais complexas, como as CNNs, que conseguem captar melhor as relações espaciais intrínsecas aos dados de imagem.

6.2.3 Modelo Centralizado - Rede neural profunda

O modelo foi composto por uma camada de entrada que recebe as imagens de 28x28, que são achatadas para vetores, seguida por múltiplas camadas densas, começando com 1024 neurônios, reduzindo gradativamente para 512, 256 e 128 neurônios. Além disso, foi implementada uma técnica de dropout de 50% após a primeira camada densa, visando reduzir o overfitting. As camadas ocultas utilizam a função de ativação ReLU e a camada final utiliza softmax, apropriada para classificação multiclasse.

Durante o treinamento, que ocorreu ao longo de 50 épocas, observamos uma evolução gradual da acurácia. Na primeira época, o modelo atingiu uma acurácia de 73,94%,

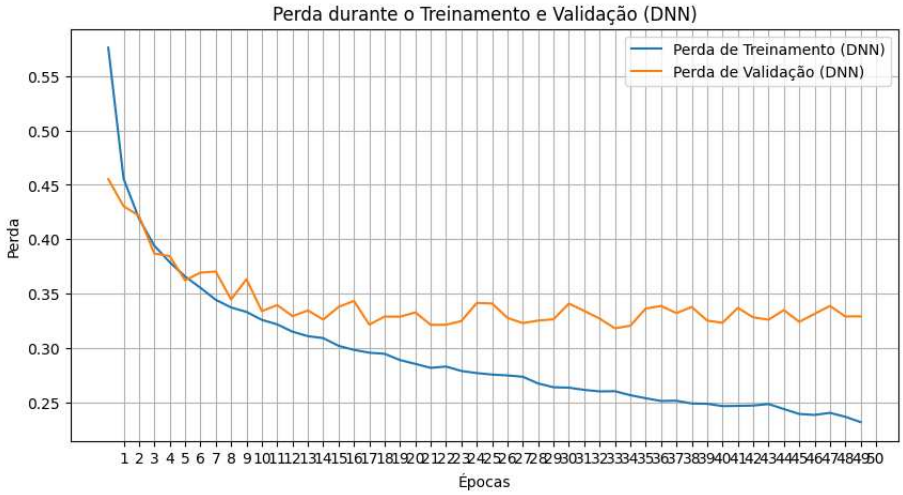
Figura 9 – Curvas de acurácia do modelo centralizado - DNN



Fonte: Elaborado pelo autor.

e ao final do treinamento, a acurácia nos dados de teste alcançou 89,69% (Figura 9). O modelo demonstrou boa capacidade de generalização para os dados de teste, indicando que foi capaz de captar os padrões das imagens de roupas e calçados presentes na base Fashion-MNIST. A perda de validação também apresentou uma tendência de queda ao longo do treinamento, apesar de ter flutuado um pouco após a 30ª época, sugerindo um leve overfitting que pode ser melhorado com ajustes de hiperparâmetros ou técnicas de regularização mais avançadas, como pode ser observado na Figura 10.

Figura 10 – Curvas de perda do modelo centralizado - DNN



Fonte: Elaborado pelo autor.

Em termos de tempo de treinamento, o modelo DNN exigiu aproximadamente 64 minutos (3856 segundos) para ser treinado, o que reflete a complexidade da arquitetura e o tempo necessário para ajustar os parâmetros com uma taxa de aprendizado reduzida

para melhorar a convergência. Comparando com os modelos mais simples como o MLP, o DNN obteve um desempenho de acurácia superior, devido à sua maior capacidade de representação e número de parâmetros. No entanto, em termos de tempo, o DNN foi mais lento que os outros modelos centralizados, o que era esperado, dada a profundidade da rede e o número de camadas e neurônios envolvidos.

6.3 Resultados do Modelo Federado

O modelo federado criado para este projeto foi projetado para treinar uma Rede Neural Convolucional (CNN) em um cenário distribuído utilizando dados descentralizados da base Fashion-MNIST. A CNN possui duas camadas convolucionais seguidas de pooling, uma camada totalmente conectada de 128 neurônios e, por fim, uma camada de saída com softmax para as 10 classes do conjunto de dados. A arquitetura foi escolhida para equilibrar a complexidade com o desempenho, enquanto a estratégia federada permitiu o treinamento local em múltiplos clientes sem centralizar os dados. As métricas analisadas incluem acurácia, acurácia top-3 e perda ao longo de 20 rodadas de treinamento.

6.3.1 Acurácia por Rodada

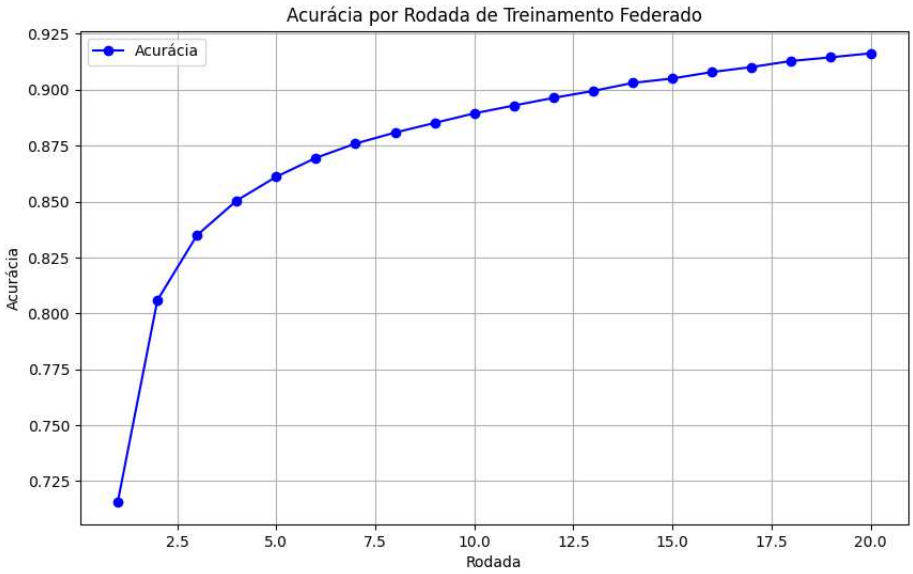
A acurácia categórica medida ao longo das rodadas indica a porcentagem de previsões corretas do modelo considerando apenas a classe mais provável. Na primeira rodada, a acurácia inicial foi de 71,58%, o que já era uma base sólida considerando o treinamento distribuído e a variedade dos dados em diferentes dispositivos. À medida que o treinamento avançou, a acurácia foi aumentando, chegando a 91,62% na vigésima rodada. A expectativa era que o modelo federado convergisse para uma acurácia próxima à obtida por um modelo centralizado, mas com desafios como a variabilidade nos dados de diferentes clientes. A acurácia por rodada é apresentada na Figura 11.

A melhoria constante ao longo das rodadas demonstra que, apesar das limitações inerentes ao aprendizado federado, a rede neural conseguiu aprender de maneira eficiente. Entretanto, ajustes finos, como a otimização dos parâmetros e o balanceamento dos dados entre os clientes, poderiam ainda mais aprimorar o desempenho.

6.3.2 Acurácia *Top-3*

A utilização da métrica de acurácia top-3 em um modelo simples, que à primeira vista não parece exigir tal métrica, pode gerar questionamentos. No entanto, há uma justificativa técnica e prática para trazê-la ao contexto de modelos de aprendizado federado, mesmo em cenários menos complexos. A acurácia tradicional, ou top-1, avalia se a previsão feita pelo modelo corresponde exatamente à classe correta. Já a acurácia top-3

Figura 11 – Curvas de acurácia do modelo federado



Fonte: Elaborado pelo autor.

expande essa perspectiva, permitindo que a previsão correta seja considerada válida se estiver entre as três principais respostas dadas pelo modelo.

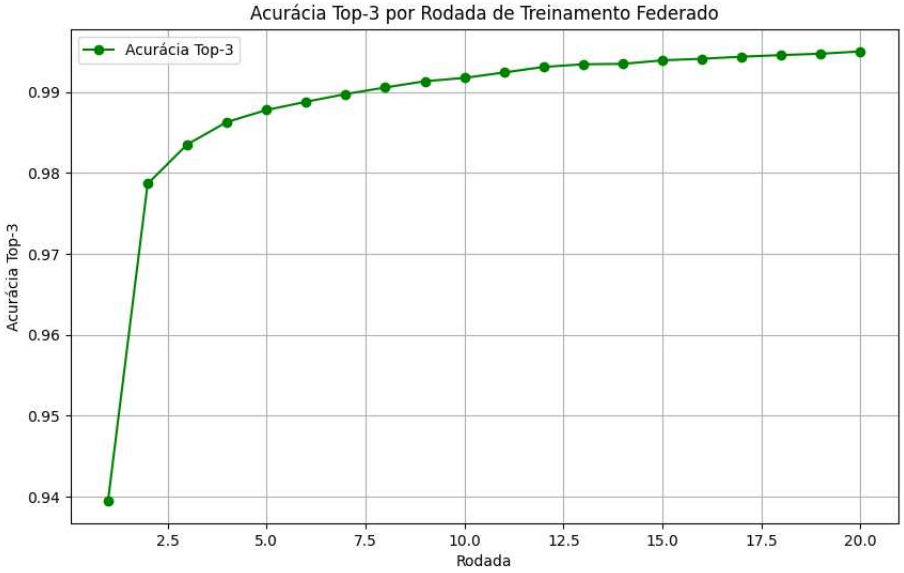
Por que então aplicar a acurácia top-3 em um modelo simples? A resposta está na capacidade dessa métrica de fornecer uma visão mais abrangente da capacidade do modelo em capturar relações próximas entre classes. Quando o modelo lida com dados que possuem classes muito similares entre si – como imagens no conjunto de dados CIFAR-10 ou MNIST – pode ser difícil determinar com precisão se um objeto pertence a uma classe específica, especialmente em casos onde diferentes classes compartilham características visuais ou padrões complexos. Assim, a acurácia top-3 ajuda a revelar se o modelo está, de fato, fazendo previsões significativas, mesmo que a primeira escolha não seja exatamente correta.

Em um cenário real, como em uma aplicação de reconhecimento de imagem em smartphones, o dispositivo pode estar treinando com imagens locais que variam significativamente das imagens em outros dispositivos. A acurácia top-3 se torna útil aqui porque permite observar se o modelo está, ao menos, identificando as classes mais próximas de forma coerente, mesmo que em alguns casos não acerte exatamente a classe correta no top-1. Isso é especialmente importante quando há variações nos dados ou ruído nas entradas.

A acurácia top-3 reflete a capacidade do modelo de incluir a classe correta entre as três principais previsões feitas. Esta métrica é particularmente útil em cenários onde a classificação pode ter ambiguidade e uma das classes mais prováveis pode ser a correta. No modelo federado, a acurácia top-3 começou em 93,95% e subiu para 99,50% na rodada

final. A acurácia top-3 é apresentada na Figura 12.

Figura 12 – Curvas de acurácia *Top-3* do modelo federado



Fonte: Elaborado pelo autor.

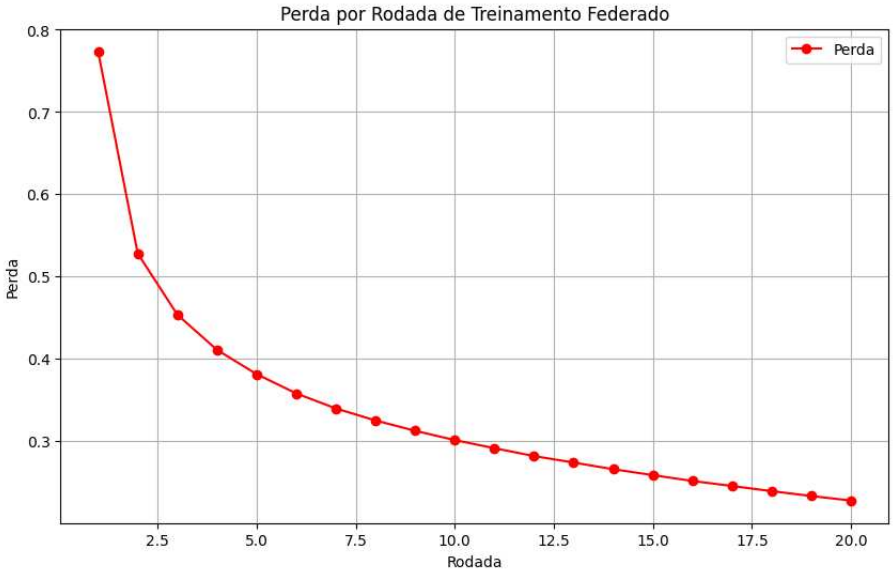
Esse resultado sugere que o modelo federado pode ser particularmente eficaz em cenários onde a precisão absoluta de top-1 pode não ser suficiente, como em recomendações de produtos ou sistemas de detecção com baixa margem de erro.

6.3.3 Perda por Rodada

A perda foi monitorada como uma métrica que quantifica o erro do modelo, sendo uma medida crítica durante o processo de otimização. Na primeira rodada, a perda era relativamente alta, com um valor de 0,773, o que reflete a incerteza inicial do modelo ao fazer previsões em um cenário federado. Ao longo do treinamento, houve uma queda consistente na perda, que foi reduzida para 0,227 ao final da vigésima rodada. A perda por rodada é apresentada na Figura 13.

A expectativa era de que a perda diminuísse à medida que o modelo ajustasse seus pesos com base nas atualizações federadas, embora a velocidade da convergência possa ser impactada pela comunicação entre os dispositivos e a distribuição não homogênea dos dados. A redução da perda confirma que o modelo aprendeu de maneira estável ao longo das rodadas, e, embora o treinamento federado tenha suas complexidades, ele foi capaz de atingir um desempenho competitivo com modelos centralizados.

Figura 13 – Curvas de perda do modelo federado



Fonte: Elaborado pelo autor.

6.4 Desempenho entre os Modelos

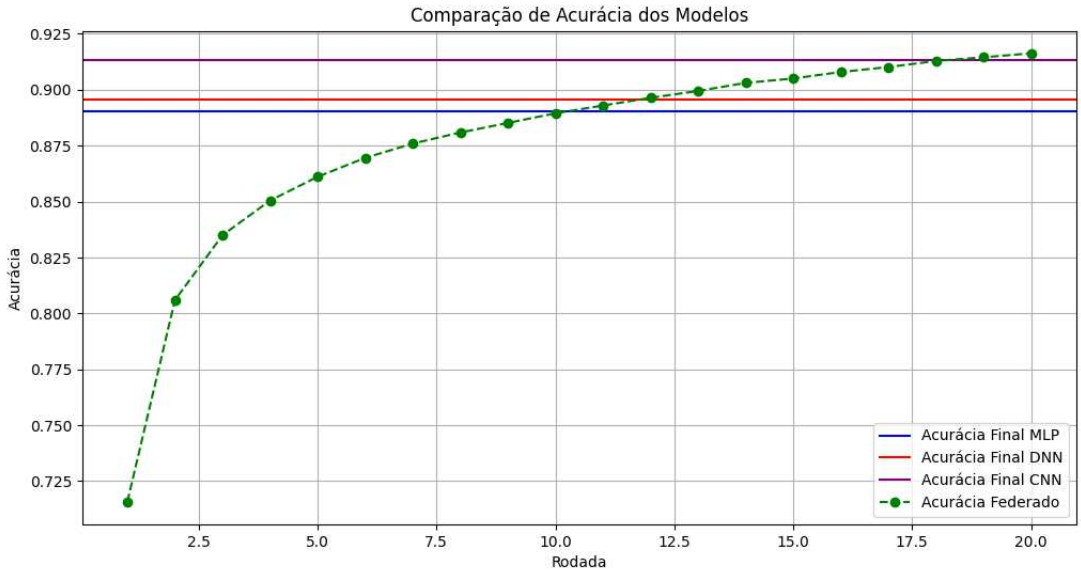
Nesta seção, analisamos o desempenho dos modelos centralizados e federados aplicados ao conjunto de dados Fashion-MNIST. As métricas de avaliação utilizadas incluem acurácia, acurácia *Top-3* e perda. A comparação entre o modelo de aprendizado federado e os modelos centralizados oferece insights valiosos sobre a eficácia e a eficiência das diferentes abordagens.

6.4.1 Acurácia

A acurácia é a principal métrica usada para avaliar a proporção de previsões corretas feitas pelo modelo em relação ao total de previsões. Nos modelos centralizados desenvolvidos com diferentes técnicas de treinamento (CNN, MLP, DNN), a acurácia foi consistentemente alta, com a CNN alcançando uma acurácia de 91,4% após 50 épocas de treinamento, o MLP atingindo 89,0%, e o DNN registrando 89,6%. Esses números refletem um desempenho sólido, especialmente considerando o uso de um conjunto de dados como o Fashion-MNIST. O modelo federado, por outro lado, apresentou uma acurácia final de 91,6% após 20 rodadas de treinamento, o que está alinhado com as expectativas de desempenho. A comparação da acurácia dos modelos Centralizados e Federado é apresentada na Figura 14.

Esperava-se que o modelo federado atingisse níveis comparáveis aos modelos centralizados, apesar das limitações inerentes à descentralização dos dados e à variação entre os dispositivos. De maneira geral, a acurácia do modelo federado mostrou-se competitiva,

Figura 14 – Comparação de acurácia entre os modelos centralizados e federados



Fonte: Elaborado pelo autor.

evidenciando que a abordagem distribuída pode ser eficaz em cenários reais sem sacrificar significativamente a precisão.

6.4.2 Perda

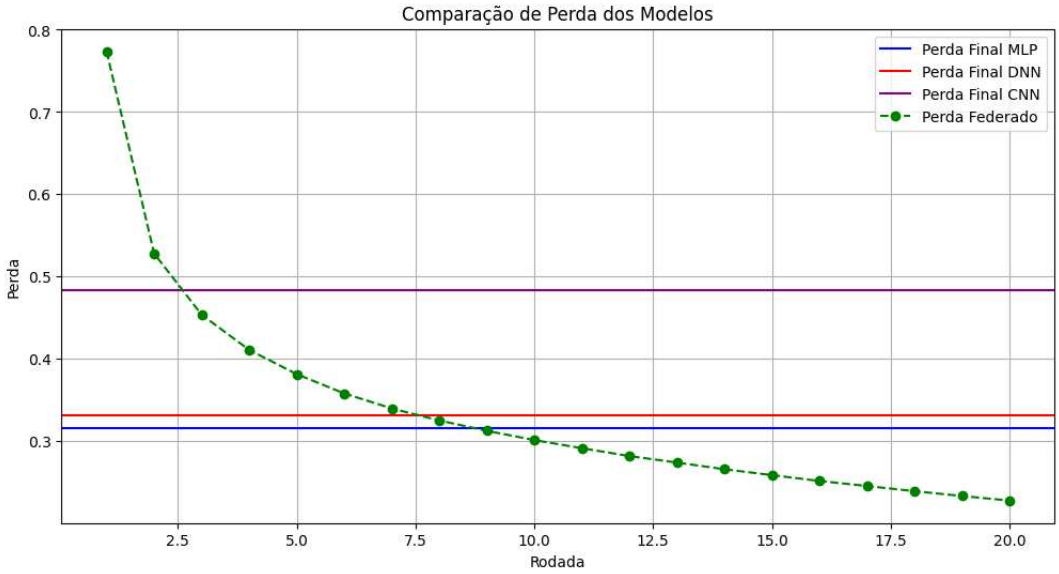
A perda é uma métrica que reflete o erro acumulado do modelo ao longo do treinamento. Em modelos de classificação como os utilizados neste estudo, a perda é geralmente expressa através da função de erro de entropia cruzada. Para os modelos centralizados, a perda final foi de 0,48 para a CNN, 0,31 para o MLP e 0,33 para o DNN, indicando que o erro foi bem minimizado ao longo do processo de treinamento. O modelo federado, por sua vez, apresentou uma perda de 0,23 na última rodada, sugerindo que ele foi capaz de reduzir o erro de maneira eficiente, mesmo em um cenário onde os dados estão distribuídos e não centralizados. Como pode ser observado na Figura 15.

Antes do treinamento, a expectativa era de que a perda no modelo federado fosse ligeiramente maior do que nos modelos centralizados, devido à complexidade adicional introduzida pela comunicação entre dispositivos e pelas diferenças nos dados locais. No entanto, o resultado final foi positivo, com a perda do modelo federado sendo inferior em algumas rodadas, o que indica que o processo federado conseguiu aprender de forma eficiente, mesmo com as restrições do ambiente distribuído.

6.4.3 Acurácia *Top-3*

A acurácia top-3 mede a capacidade do modelo de prever a classe correta dentro das três classes mais prováveis, o que é especialmente útil em cenários onde há ambiguidades

Figura 15 – Comparação de perda entre os modelos centralizados e federados



Fonte: Elaborado pelo autor.

entre classes. Para os modelos centralizados, essa métrica não foi usada inicialmente, mas foi incluída no modelo federado como uma forma de explorar cenários mais realistas de classificação. A acurácia top-3 do modelo federado atingiu 99,5% na última rodada, o que significa que, mesmo em casos onde o modelo não acertou a classe exata, ele foi capaz de incluir a resposta correta entre suas três principais previsões na maioria das vezes.

Isso era esperado, uma vez que a acurácia top-3 geralmente tende a ser mais alta do que a acurácia categórica, dado que o modelo tem mais chances de acertar dentro de um intervalo maior de possibilidades. A alta acurácia top-3 no modelo federado sugere que ele poderia ser altamente eficaz em cenários de produção, onde a precisão absoluta nem sempre é o critério mais importante.

6.5 Considerações Finais

Comparando o desempenho geral dos modelos, pode-se observar que o modelo federado apresentou desempenho semelhante, e em alguns casos, até superior em relação aos modelos centralizados em termos de acurácia e perda. A principal vantagem do aprendizado federado é a preservação da privacidade dos dados, mas isso geralmente vem com o custo de maior complexidade no treinamento e maiores exigências computacionais. No entanto, os resultados demonstram que, com as devidas otimizações, o aprendizado federado pode atingir níveis de desempenho muito próximos dos modelos tradicionais centralizados, sem comprometer significativamente a precisão ou aumentar excessivamente a perda. A inclusão da acurácia top-3 também destaca o potencial do aprendizado federado para ser utilizado em cenários de classificação mais complexos, onde várias respostas podem ser

válidas.

Em termos de expectativas, esperava-se que os modelos centralizados se saíssem um pouco melhor em acurácia e perda, devido à sua simplicidade e eficiência computacional, mas os resultados mostraram que o modelo federado foi capaz de se manter competitivo. Essa conclusão reforça a ideia de que o aprendizado federado é uma solução viável para cenários descentralizados, onde a privacidade é uma preocupação central, sem sacrificar o desempenho.

6.6 Dificuldades e Limitações

A seção de dificuldades e limitações explora os desafios enfrentados durante o desenvolvimento dos modelos de aprendizado centralizado e federado, as influências dessas dificuldades na escolha da base de dados e as implicações do uso do Google Colab no treinamento dos modelos. Esta análise fornece uma visão crítica sobre as limitações dos métodos de treinamento e os aspectos técnicos que impactaram o progresso e os resultados do projeto.

6.6.1 Dificuldades no Desenvolvimento dos Modelos

Desenvolver modelos de aprendizado centralizado e federado envolveu uma série de desafios técnicos e conceituais. Para os modelos centralizados, como o MLP e o DNN, uma das principais dificuldades foi o ajuste dos hiperparâmetros, como a taxa de aprendizado e o número de camadas e neurônios. A escolha inadequada desses parâmetros pode levar a problemas como overfitting ou underfitting, impactando negativamente a precisão e a capacidade de generalização dos modelos. Além disso, a gestão do treinamento e a monitorização das métricas de desempenho foram essenciais para garantir que o modelo estivesse aprendendo de forma eficaz e não estivesse simplesmente decorando os dados de treinamento.

6.6.2 Influência na Escolha da Base de Dados

A escolha da base de dados Fashion-MNIST foi feita de forma estratégica, levando em consideração a necessidade de um dataset que representasse um cenário de classificação realista, mas que ainda fosse manejável em termos computacionais. Fashion-MNIST apresenta um desafio maior do que o MNIST tradicional, mas ainda assim é uma base de dados controlada e de tamanho razoável, o que facilitou a comparação de resultados entre os diferentes modelos. No entanto, apesar da simplicidade relativa dos dados, a diversidade das imagens de vestuário e as pequenas diferenças entre classes exigiram que os modelos fossem cuidadosamente ajustados para maximizar a precisão sem sobrecarregar os recursos computacionais disponíveis.

6.6.3 Impacto do Google Colab no Treinamento dos Modelos

O uso do Google Colab teve um impacto no desenvolvimento e no treinamento dos modelos. Google Colab fornece uma plataforma acessível e com recursos computacionais relativamente poderosos, como GPUs, o que facilitou o treinamento dos modelos, especialmente para o modelo DNN que exige maior capacidade computacional. No entanto, existem limitações associadas ao uso de Google Colab, como restrições de tempo de execução e limitações de memória. Essas restrições podem impactar o treinamento de modelos mais complexos ou o uso de grandes conjuntos de dados.

6.6.4 Limitações dos Métodos de Treinamento

Os métodos de treinamento centralizado e federado apresentam suas próprias limitações. No treinamento centralizado, um desafio é a necessidade de grandes volumes de dados e poder computacional para alcançar resultados precisos e generalizáveis. Modelos mais complexos podem exigir ajustes finos e otimização que podem não ser práticos em todas as situações. Além disso, o treinamento centralizado não aborda diretamente a questão da privacidade dos dados, o que pode ser uma limitação significativa em contextos onde a proteção da privacidade é essencial.

Por outro lado, o aprendizado federado, embora ofereça vantagens significativas em termos de privacidade e segurança dos dados, enfrenta limitações relacionadas à comunicação e sincronização entre os clientes. A agregação de modelos locais pode ser influenciada pela heterogeneidade dos dados e pelas condições variáveis de treinamento nos diferentes clientes. Além disso, o custo computacional e a complexidade do protocolo de treinamento federado podem ser desafiadores, especialmente quando se lida com um grande número de clientes e uma vasta quantidade de dados.

7 Conclusao

Esse capítulo possui como intenção apresentar as considerações finais sobre esse trabalho, considerando a solução desenvolvida e os retornos referentes à utilização da solução, além de retomar as questões e objetivos inicialmente especificados no Capítulo 1. Inicialmente, há um breve resumo sobre o Contexto Geral definido para o trabalho e reflexões sobre o mesmo. Após isso, será exposto sobre o status atual do trabalho, retomando Questionamentos e Objetivos. Há destaque ainda para as Contribuições e Fragilidades da solução apresentada. Por fim, constam ideias para Trabalhos Futuros, visando evoluções para a solução atual.

7.1 Contexto Geral

O avanço das tecnologias de aprendizado de máquina e inteligência artificial tem proporcionado grandes benefícios, mas também levantado questões sobre como os dados dos usuários são coletados, armazenados e utilizados. O aprendizado federado surge como proposta de solução, permitindo o treinamento de modelos de IA em um ambiente distribuído sem a necessidade de centralizar os dados. Este método não só garante que os dados permaneçam na origem, preservando a privacidade dos usuários, mas também melhora a eficácia dos modelos ao aproveitar a diversidade dos dados distribuídos. A motivação para explorar este tema reside na necessidade de encontrar um equilíbrio entre a inovação tecnológica e o respeito pela privacidade, um tema de relevância crescente na sociedade atual.

Sabendo disso, surgiu o interesse de explorar o aprendizado federado em um contexto prático, com o objetivo de entender melhor suas vantagens e desafios, assim como as possíveis aplicações em cenários reais e discussões acerca da aplicabilidade do modelo e as propostas que ele carrega consigo. A solução desenvolvida foi testada em um cenário de classificação de imagens de dígitos manuscritos, utilizando o conjunto de dados Fashion-MNIST. Os resultados obtidos foram promissores, demonstrando a viabilidade do aprendizado federado para a tarefa proposta.

7.2 Questionamentos e Objetivos

A seguir, são retomadas as questões inicialmente apresentadas no trabalho, assim como os objetivos específicos e geral especificados para serem cumpridos ao longo do projeto.

7.2.1 Questão de Pesquisa

Durante a confecção do projeto, comparamos o desempenho do modelo federado com modelos centralizados, como MLP e DNN, utilizando a base de dados Fashion-MNIST. Os resultados demonstraram que o modelo federado conseguiu manter uma performance competitiva em termos de acurácia e perda, semelhante aos modelos centralizados, apesar de operar em um cenário descentralizado.

Além de seu desempenho, o aprendizado federado é particularmente eficaz em preservar a privacidade dos usuários. A privacidade é protegida de várias maneiras. Primeiramente, os dados nunca saem dos dispositivos dos usuários, o que minimiza o risco de vazamentos ou acessos não autorizados. Em segundo lugar, a agregação dos gradientes ao invés dos dados brutos dificulta a recuperação de informações pessoais individuais. Adicionalmente, técnicas de privacidade como a *differential privacy* podem ser aplicadas para adicionar ruído às atualizações do modelo, tornando ainda mais difícil a inferência sobre dados específicos a partir dos gradientes.

Nosso estudo confirmou que o aprendizado federado não só oferece uma alternativa viável e eficaz para o treinamento de IA em ambientes descentralizados, mas também é uma abordagem sólida para garantir a privacidade dos usuários. Isso demonstra que, embora o aprendizado federado enfrente desafios técnicos, ele oferece uma solução equilibrada entre performance e proteção de dados, alinhando-se com as crescentes demandas por privacidade e segurança na era digital.

7.2.2 Questão de Desenvolvimento

Um dos principais desafios na implementação prática do aprendizado federado é a eficiência computacional. A descentralização do treinamento implica que cada dispositivo cliente deve processar e treinar modelos localmente, o que pode ser oneroso em termos de recursos computacionais e energia. Em nosso projeto, observou-se que modelos mais complexos, como o DNN (Deep Neural Network), demandam significativamente mais tempo e recursos para treinamento comparado a modelos mais simples. Além disso, o processo de agregação de gradientes no servidor central requer um balanceamento cuidadoso entre a carga computacional distribuída e a capacidade do servidor de processar e combinar esses gradientes eficientemente. A combinação dessas demandas pode resultar em um tempo de treinamento mais longo e maior uso de recursos em comparação com métodos centralizados, onde o processamento é concentrado em um único servidor.

Outro desafio crítico é a comunicação entre dispositivos. Em um ambiente federado, os dispositivos devem trocar informações sobre os gradientes ou atualizações do modelo com o servidor central em intervalos regulares. Isso pode ser problemático devido à largura de banda limitada e à latência da rede. No contexto do nosso estudo, foi observado que o

treinamento federado em redes com largura de banda reduzida pode levar a um aumento no tempo total de treinamento e a uma menor eficiência na sincronização dos modelos. A necessidade de enviar atualizações frequentes e de alta frequência pode ser um gargalo, especialmente em cenários com muitos clientes ou em redes de baixa qualidade.

A adaptação a diferentes tipos de dados é outro desafio. O AF assume que os dados distribuídos entre os clientes podem variar em termos de distribuição e qualidade. No entanto, em cenários do mundo real, onde os dados podem ser altamente variados e desbalanceados, a eficácia do aprendizado federado pode ser comprometida. Isso pode levar a problemas como a convergência lenta do modelo e uma menor precisão geral, já que o modelo precisa ser capaz de generalizar bem sobre uma vasta gama de tipos e qualidades de dados. Estratégias para lidar com a heterogeneidade dos dados, como técnicas de personalização de modelos e ajuste de hiperparâmetros, são essenciais para superar essas limitações.

Portanto, a questão de pesquisa revelou que, embora o aprendizado federado ofereça uma abordagem promissora para o treinamento descentralizado de modelos de IA, ele enfrenta desafios relacionados à eficiência computacional, comunicação entre dispositivos e adaptação a diferentes tipos de dados. Superar essas dificuldades é crucial para a adoção mais ampla do AF em aplicações práticas, e futuras pesquisas e desenvolvimentos são necessários para otimizar essas áreas e melhorar a viabilidade e a eficácia da abordagem federada.

7.2.3 Objetivos Alcançados

Foram especificacados os seguintes objetivos para o projeto:

- Objetivo específico 1: Conceituar Aprendizado federado e apontar as estratégias de aprendizado de máquina que podem ser usadas para preservação da privacidade dos usuários e de dados sensíveis. Status: Cumprido, apresentado no Capítulo 1.
- Objetivo específico 2: Discorrer sobre técnicas de segurança e privacidade necessárias para proteger os dados dos usuários e o provedor do serviço durante o processo de treinamento. Status: Cumprido, apresentado no Capítulo 1.
- Objetivo específico 3: Analisar o problema de uso de dados decentralizados, identificando os principais desafios. Status: Cumprido, apresentado no Capítulo 1, 5 e 6.
- Objetivo específico 4: Comparar a eficácia do modelo de treinamento descentralizado quando comparado com modelos tradicionais de treinamento. Status: Cumprido, apresentado no Capítulo 5 e 6.

Atingindo-se os objetivos específicos, também é possível analisar o objetivo geral do trabalho, que era o investigar como o treinamento de IA's por meio do aprendizado federado pode ser efetivamente aplicado no contexto de dados descentralizados.

7.3 Contribuições e Fragilidades

A comparação entre os modelos de aprendizado federado e os modelos centralizados (CNN, MLP e DNN) revelou não apenas a viabilidade do aprendizado federado, mas também suas vantagens em termos de preservação da privacidade dos dados dos usuários. O trabalho também contribuiu para a compreensão dos desafios específicos associados ao treinamento de modelos em um ambiente descentralizado, incluindo problemas de eficiência computacional e comunicação entre dispositivos. O desenvolvimento de um modelo federado que foi comparado com modelos centralizados permitiu avaliar as trade-offs entre privacidade e desempenho, oferecendo insights valiosos para futuras pesquisas e aplicações práticas.

No entanto, o projeto também revelou algumas fragilidades que merecem consideração. Primeiramente, a eficiência computacional do aprendizado federado apresentou desafios notáveis. O tempo de treinamento do modelo federado foi consideravelmente maior em comparação com os modelos centralizados, evidenciando a necessidade de otimização dos processos de comunicação e agregação. A alta latência associada à comunicação entre dispositivos e a complexidade do algoritmo de agregação impactaram negativamente o desempenho do treinamento.

Além disso, a heterogeneidade dos dados distribuídos apresentou um desafio adicional. A variabilidade nos dados dos clientes pode influenciar negativamente a performance do modelo federado, tornando-o menos eficaz em termos de generalização. Apesar dos avanços na técnica de federated averaging, a capacidade do modelo para lidar com diferentes tipos de dados ainda é limitada, o que pode comprometer a sua eficácia em cenários do mundo real.

Finalmente, a dependência de plataformas como Google Colab para o treinamento dos modelos trouxe algumas limitações práticas. A execução em ambientes compartilhados pode limitar a flexibilidade e o controle sobre os recursos computacionais, impactando a escalabilidade e a eficiência do processo de treinamento. Estes fatores precisam ser endereçados em futuras implementações para melhorar a viabilidade prática do aprendizado federado em cenários descentralizados.

7.4 Trabalhos Futuros

Considerando as fragilidades apontadas, há espacço para futuros trabalhos referentes à solução apresentata, conforme segue:

- Otimização da eficiência computacional: Investigar estratégias para reduzir o tempo de treinamento e a carga computacional associada ao aprendizado federado, incluindo a otimização dos algoritmos de comunicação e agregação.
- Adaptação a diferentes tipos de dados: Explorar técnicas para lidar com a heterogeneidade dos dados distribuídos, incluindo a personalização de modelos e o ajuste de hiperparâmetros para melhorar a generalização e a precisão do modelo federado.
- Implementação em ambientes práticos: Desenvolver uma implementação prática do aprendizado federado em um cenário do mundo real, considerando as limitações e desafios associados à execução em ambientes compartilhados.
- Extração das métricas de validação do modelo federado: Durante o desenvolvimento deste projeto a função que realizava a extração das métricas de validação do modelo federado foi descontinuada, se fazendo necessário estudar outras formas de realizar o cálculo dessas métricas para que se tenha um maior entendimento a respeito do desempenho do modelo federado.
- Avaliação de métricas adicionais: Investigar métricas adicionais para avaliar o desempenho do modelo federado, incluindo a eficiência de comunicação, a escalabilidade e a robustez do modelo em cenários com alta variabilidade de dados.

Referências

AARONSON, S. Read the fine print. *Nature*, v. 531, p. 169–171, 2016. Citado na página 29.

ABADI, M.; AL. et. Deep learning with differential privacy. In: *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. [S.l.: s.n.], 2016. Citado na página 26.

BAGDASARYAN, E.; VEJDEMO-JOHANSSON, M.; SHMATIKOV, V. How to backdoor federated learning. *arXiv preprint arXiv:1807.00459*, 2018. Citado na página 27.

BBC. Strava fitness app 'reveals military bases'. 2018. Disponível em: <https://www.bbc.com/news/technology-42853072>. Citado na página 16.

BISHOP, C. M. *Pattern Recognition and Machine Learning*. [S.l.]: Springer, 2006. Citado na página 22.

BLOOMBERG. Amazon workers are listening to what you tell alexa. 2019. Disponível em: <https://www.bloomberg.com/news/articles/2019-04-10/is-anyone-listening-to-you-on-alexa-a-global-team-reviews-audio>. Citado na página 16.

BONAWITZ, K. et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019. Citado na página 25.

DWORK, C. A firm foundation for private data analysis. *Communications of the ACM*, 2011. Citado 2 vezes nas páginas 25 e 26.

DWORK, C. et al. Calibrating noise to sensitivity in private data analysis. *Journal of Privacy and Confidentiality*, v. 7, n. 3, p. 3–14, 2006. Citado na página 14.

EduTechCollab. *Transforming Education with Federated Learning: A Collaborative Case Study*. 2020. <https://www.edutechcollab.org/case-studies/federated-learning-education>. Citado na página 27.

EnergyTechCollab. *Decentralized Demand Forecasting in the Energy Sector using Federated Learning*. 2019. <https://www.energytechcollab.org/case-studies/federated-learning-energy>. Citado na página 28.

FLORIDI, L.; COWLS, J. A unified framework of five principles for ai in society. *Harvard Data Science Review*, v. 1, n. 1, 2018. Citado na página 22.

FREDRIKSON, M.; JHA, S.; RISTENPART, T. Model inversion attacks that exploit confidence information and basic countermeasures. In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. [S.l.: s.n.], 2014. Citado 2 vezes nas páginas 22 e 26.

GEYER, R. C. et al. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017. Citado 2 vezes nas páginas 23 e 26.

GIL, A. *Como Elaborar Projetos de Pesquisa*. [S.l.]: Atlas, 2002. v. 4. Citado na página 19.

GIOVANNI, M.; AL. et. Post-quantum cryptography: A comprehensive review. *IEEE Transactions on Quantum Engineering*, v. 1, p. 1–25, 2020. Citado na página 29.

GOH, G. Why do deep learning networks scale? *arXiv preprint arXiv:1710.06168*, 2017. Citado na página 22.

GOODFELLOW. *Deep Learning*. [S.l.]: MIT Press, 2016. Citado 2 vezes nas páginas 21 e 23.

GUARDIAN, T. The cambridge analytica files. 2018. Disponível em: <https://www.theguardian.com/news/series/cambridge-analytica-files>. Citado na página 16.

HINTON, G. E. et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, v. 29, n. 6, p. 82–97, 2012. Citado na página 21.

HITAJ. Deep models under the gan: Information leakage from collaborative deep learning. In: ACM. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. [S.l.], 2017. p. 603–618. Citado na página 16.

JONES, A.; AL. et. Federated learning for fraud detection in financial institutions. *Journal of Financial Technology*, v. 15, n. 2, p. 45–62, 2020. Citado na página 27.

KONEčNý, J.; AL. et. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016. Citado na página 28.

LI, Q. et al. A survey on federated learning systems: Vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, 2021. Citado na página 15.

LI, T. et al. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, IEEE, v. 37, n. 3, p. 50–60, 2020. Citado na página 15.

LI, Z.; SHARMA, V.; MOHANTY, S. P. Preserving data privacy via federated learning: Challenges and solutions. *IEEE Consumer Electronics Magazine*, IEEE, 2020. Citado na página 15.

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. Citado 5 vezes nas páginas 14, 15, 23, 24 e 25.

MOHASSEL, P.; AL. et. Secureml: A system for scalable privacy-preserving machine learning. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017. Citado na página 29.

ONE, C. Capital one data breach. 2019. Disponível em: <https://www.capitalone.com/facts2019/>. Citado na página 16.

PrivacyInTech. *Balancing Personalization and Privacy: A Social Media Federated Learning Approach*. 2022. <https://www.privacyintech.org/case-studies/social-media-federated-learning>. Citado na página 28.

QAYYUM, A. et al. Collaborative federated learning for healthcare: Multi-modal covid-19 diagnosis at the edge. *IEEE Open Journal of the Computer Society*, IEEE, v. 3, p. 172–184, 2022. Citado na página 15.

RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. [S.l.]: Pearson, 2016. Citado na página 22.

RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 4th. ed. Upper Saddle River, NJ: Pearson, 2022. Citado 2 vezes nas páginas 13 e 14.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, v. 3, n. 3, p. 210–229, 1959. Citado na página 21.

SciCollab. *Advancing Scientific Collaboration with Federated Learning*. 2021. <https://www.scicollaboration.org/case-studies/federated-learning-scientific-collaboration>. Citado na página 28.

SHOKRI, R.; AL. et. Membership inference attacks against machine learning models. In: *IEEE Symposium on Security and Privacy*. [S.l.: s.n.], 2015. Citado na página 26.

SHOKRI, R.; SHMATIKOV, V. Privacy-preserving deep learning. *arXiv preprint arXiv:1412.6074*, 2015. Citado 2 vezes nas páginas 15 e 27.

SMITH, J.; AL. et. Privacy-preserving diagnostics: A federated learning approach in healthcare. *Journal of Medical Informatics*, v. 23, n. 4, p. 112–129, 2018. Citado na página 27.

SUTSKEVER, I. et al. On the importance of initialization and momentum in deep learning. In: *Proceedings of the 30th International Conference on Machine Learning*. [S.l.: s.n.], 2013. Citado na página 23.

UNION, E. Diretiva de proteção de dados pessoais da união europeia. 2016. Disponível em: <https://commission.europa.eu/law/law-topic/data-protection/data-protection-eu_pt>. Citado na página 14.

YANG, Q. et al. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, v. 10, n. 2, p. 1–19, 2019. Citado 2 vezes nas páginas 23 e 25.