UNIVERSIDADE FEDERAL DO PARANÁ

BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO

FELIPE DELLA GIUSTINA

Exploração de Aprendizagem Federada com CNN

Curitiba

2024

FELIPE DELLA GIUSTINA

Exploração de Aprendizagem Federada com CNN

Trabalho apresentado à Trabalho de Conclusão de Curso em Bacharelado em Ciência da Computação da Universidade Federal do Paraná.

Orientadora: Prof. Dra. Aurora Trinidad Ramirez Pozo

Curitiba

2024

Resumo

O crescimento de redes IoT e a crescente necessidade de trabalhar com dados e problemas cada vez mais volumosos e complexos demandam uma mudança de paradigma. As abordagens centralizadas estão chegando ao seu limite, o que impulsiona o surgimento de novas alternativas descentralizadas. Problemas complexos se tornam manejáveis com o uso de aprendizado de máquina em contextos descentralizados, abrindo novos horizontes. A aprendizagem Federada é a aplicação de aprendizado de máquina em contexto descentralizado, que fornece opções de segurança, privacidade e processamento de dados descentralizado. Este trabalho apresentará conceitos básicos sobre Redes Neurais Convolucionais e aprendizagem federada, além de abordar vantagens e limitações de se usar esse tipo de aprendizado de máquina. Além disso, serão apresentadas aplicações reais desse tipo de abordagem em diversos ambientes, como domicílios, IoT, grandes empresas e drones. Por fim, este trabalho inclui a apresentação de um exemplo prático de uma implementação de aprendizagem federada utilizando o Flower Framework e o banco de dados CIFAR-10. Nela são analisadas os seus parâmetros de aprendizagem federada e como as suas variações para avaliação de que fatores influenciam no comportamento da aprendizagem com o uso de aprendizagem federada.

Palavras chave: Aprendizagem Federada; Aprendizado de Máquina; Rede Neural Convolucional; CIFAR-10; Flower Framework.

Abstract

The growth of IoT networks with the growing need to work with increasingly large and complex data and problems has demanded a paradigm shift to alternatives. Centralized approaches are reaching their limit, which drives the emergence of new decentralized alternatives. Complex problems become manageable with the use of machine learning in decentralized contexts, opening new horizons. Federated learning is the application of machine learning in a decentralized context, which provides security, privacy, and decentralized data processing options. This work will present basic concepts about Convolutional Neural Networks and federated learning, in addition to addressing advantages and limitations of using this type of machine learning. In addition, this work demonstrates real applications of this type of approach in various environments, such as homes, IoT, large companies, and drones. Finally, this work includes the presentation of a practical example of a federated learning implementation using the Flower Framework and the CIFAR-10 database. It analyzes its federated learning parameters and their variations to assess which factors influence learning behavior with the use of federated learning.

Keywords: Federated Learning; Machine Learning; Convolutional Neural Network; CIFAR-10; Flower Framework.

Lista de Figuras

LISTA DE SIGLAS

CNN	Rede Neural Convolucional /*Convolutional neural network*
FedAvg	*Federated Averaging*
FF	Framework Flower
FL	Aprendizagem Federada /*Federated Learning*
ML	*Machine learning*
MLP	Rede multi-camada de perceptrons/*Multiple Layers Perceptron*
RN	Rede Neural

Índice

1. Introdução

O crescimento de redes IoT e a crescente necessidade de trabalhar com dados e problemas cada vez mais volumosos e complexos demandam uma mudança de paradigma. As abordagens centralizadas estão chegando ao seu limite, o que impulsiona o surgimento de novas alternativas descentralizadas. Problemas complexos se tornam manejáveis com o uso de aprendizado de máquina em contextos descentralizados, abrindo novos horizontes.

A Aprendizagem Federada(FL ou *Federated Learning*) é a aplicação de aprendizado de máquina em contexto descentralizado, que fornece opções de segurança, privacidade e processamento de dados descentralizado. Este trabalho apresentará conceitos básicos sobre Redes Neurais Convolucionais(CNN ou *Convolutional neural network*) e aprendizagem federada, além de abordar vantagens e limitações de se usar esse tipo de aprendizado de máquina. Além disso, será apresentado aplicações reais desse tipo de abordagem em diversos ambientes, como domicílios, IoT, grandes empresas e drones. Variações de FL também serão apresentadas ao longo do trabalho e exemplos de suas aplicações também serão apresentados.

Por fim, este trabalho inclui a apresentação de um exemplo prático de uma implementação de aprendizagem federada utilizando o Flower Framework(FF) e o banco de dados CIFAR-10. Nela são analisadas os seus parâmetros de aprendizagem federada e como as suas variações para avaliação de que fatores influenciam no desempenho da implementação desse tipo de aprendizagem.

2. Fundamentação teórica

A aprendizagem se dá quando um agente está melhorando o desempenho em atividades futuras baseados em observações sobre o ambiente presente (S. Russel,2013). Ao se aplicar esse conceito a programação e computadores, podemos ter um agente, ou seja, um programa, que aprende baseado em dados apresentados e que tenta encontrar uma solução.

No aprendizado de Máquina (ML ou *Machine Learning*) é necessário bases de dados extensivas para realizar a aprendizagem devida e em anos recentes os modelos de aprendizagem e bases de dados tem se tornado cada vez mais complexas (Yurdem, 2024).

A vantagem desta aproximação de programação é que esses agentes podem resolver um problema sem que o programador tenha que conhecer todas as condições presentes e futuras do ambiente, pois o próprio agente extrapola soluções baseadas no ambiente apresentado durante o treinamento.

Para que essa aprendizagem atinja um resultado satisfatório, a máquina deve "aprender"(também chamado de treinamento) a resolver um problema utilizando um conjunto de dados de treinamento, um modelo matemático que representa o aprendizado e uma métrica para avaliação do aprendizado realizado. Outras técnicas e partes existem para auxiliar a criação de modelos mais eficientes ou adaptados a situações específicas, porém essas são as partes principais do funcionamento de aprendizagem de máquina.

Neste trabalho focaremos em um tipo específico de aprendizagem de máquina chamada de Rede Neural Convolucional, que pode ser usada em visão computacional para o reconhecimento, classificação e detecção de imagens e objetos presentes em imagens. O CNN é um modelo de aprendizagem inspirado em como os neurônios funcionam no cérebro, no qual as informações de entrada são filtradas e manipuladas. Essas manipulações de dados são ajustadas a cada erro com o intuito de aproximar cada vez mais a acurácia dos resultados obtidos nos dados que saem da CNN.

Esses dados utilizados para o aprendizado são apresentados em duplas , dados de entrada e dados de saída desejada, que são divididos de forma específica para maximizar o aprendizado e validação do aprendizado (S. Russel,2013). Cada unidade de dado que entra na camada inicial possui um rótulo, com o aprendizado cada unidade de dado deve gerar seu respectivo rótulo com um certo grau de confiabilidade para ser considerado uma aprendizagem de sucesso.

Esse aprendizado é realizado por operações lógicas ao longo da rede que são afetadas pelos pesos de cada operação. Essas que nada mais são que um conjunto de funções não lineares, e cada unidade dessas funções representa o equivalente a um neurônio. Durante a fase de aprendizado cada erro é corrigido utilizando a técnica de backpropagation, isso faz com que os pesos de cada camada sejam aos poucos ajustada para se atingir o resultado ideal.

Esse ajuste necessário é calculado com a ajuda da função chamada “Loss Function” que mensura a distância entre o resultado obtido e o resultado desejado. Então utilizando funções matemáticas o ajuste é realizado camada por camada na direção contrária do processamento de dados, ou seja, da camada final se propagando até o início. Dessa forma, as mudanças e ajustes vão se acomodando ao longo de iterações até se atingir o resultado desejado.

2.1 Arquitetura de uma Rede Neural Convolucional

Antes de entendermos o que é uma CNN devemos entender o seu antecessor, o perceptron e a Rede Neural(RN). Inspirado em como um conjunto de neurônios constitui um cérebro, que é uma estrutura complexa, ao utilizar expressões lógicas e criar o equivalente a uma rede, conseguimos criar operações ainda mais complexas(McCulloch, 1943). Eventualmente esse conceito se desenvolveu ao que hoje chamamos de perceptron, uma expressão matemática que recebe múltiplos dados de entrada e produz uma saída dependendo desses dados recebidos. Se unirmos camadas de perceptrons temos um uma rede de perceptrons de múltiplas camadas(“Multiple Layers Perceptron” ou MLP). Essas redes são capazes de aprender e classificar dados e problemas mais complexos, porém redes MLP mais complexas são custosas em tempo e dados durante o treinamento, essas limitações eventualmente levaram ao desenvolvimento de outras alternativas de redes neurais, entre elas a CNN.

A CNN é um tipo de aprendizagem de máquina que, assim como o perceptron, se inspira na forma que neurônios interagem na vida real para realizar o aprendizado. Essas redes funcionam utilizando camadas de nodos, no qual nodos realizam o processamento de dados e que se organizam em camadas, essas que são conjuntos de nodos logicamente similares. Dessa forma, os dados se propagam de uma camada para a próxima passando pelos nodos de cada camada até que o resultado final seja obtido na última camada da rede. As camadas iniciais das CNN trabalham vários tipos de filtros de dados antes da aprendizagem ser realizada nas camadas de convolução, como veremos adiante.

Cada camada utiliza uma lógica diferente para extrair informação dos dados recebidos para o processamento de camadas futuras, logo mais a frente deste trabalho discutiremos as camadas mais relevantes para qualquer tipo de CNN. De qualquer modo, é importante ter uma noção geral sobre os usos de cada um dos tipos de camada de uma CNN. Com isso em mente, a seguir, há uma explicação breve sobre o uso de tipos de camadas:

- Camadas de entrada: Recebem os dados e repassam para próximas camadas
- Camada de Saída: Camada de saída dos dados, elas que fazem o processo final de classificação.
- Camadas ocultas: Camadas de uma rede que não são de entrada nem de saída.

- Camadas de Convolução: São camadas que filtram e operam nos dados para refinar as características em futuras camadas.
- Pooling: Reduz o volume de dados ao compactar dados com o uso de operações de matriz.
- Fully Connected: Camadas que possuem todos os seus nós, ou grande parte deles, conectados entre si. Nessa camada todos resultados de entrada afetam todos os resultados de saída.

Muitas vezes os dados utilizados são muito volumosos para a realização do aprendizado de forma eficiente, assim os dados são passados por “filtros”(ou “kernels”) que realizam operações de matrizes para diminuir o volume dos dados sem grande perda na qualidade dos mesmos. Outro uso é para extração de características desses dados, como por exemplo, realizar uma operação de detecção de bordas em uma imagem.

2.1.1 Camadas de entrada e saída

A camada de entrada tem como entrada os dados a serem utilizados para o aprendizado e cada nodo de entrada recebe um dos dados que serão utilizados para o aprendizado, esses dados escolhidos são chamados de características(ou “features”).

A camada de saída é onde ocorre a classificação dos dados que foram inseridos na camada de entrada. Geralmente o número de classes na base de dados apresentada pelo treinamento é o mesmo que o número de nodos de saída. Dessa forma é mais fácil avaliar e corrigir a rede durante a fase de aprendizado.

2.1.2 Camada de Pooling

Existem situações nas quais o espaço ocupado pelos dados é grande e isso necessita de uma capacidade de processamento maior que a desejada, assim, camadas de “pooling” agrupa features e repassa os dados com uma dimensionalidade menor para as próximas camadas. Esse agrupamento é realizado através de kernels, que nada mais são que matrizes que adquirem um subconjunto dos dados que entraram na camada e os agregam com o uso de operações sobre matrizes.

2.1.3 Camadas de Convolução

Depois que os dados são devidamente processados em camadas anteriores, a camada de convolução é onde ocorre a maior parte do processamento para a manipulação das características extraídas pelas camadas anteriores antes de passar os dados resultantes para a próxima camada. Nessas camadas é onde realmente inicia o processo de aprendizagem, que faz o uso de expressões matemáticas com pesos variáveis que vão se ajustando ao longo das iterações de aprendizagem da CNN. Essas Camadas costumam ser as mais densas que

eventualmente geram uma saída de nodos iguais as classificações. Por exemplo, um dataset com 12 classificações distintas teria um modelo de CNN com 12 nodos finais de saída.

2.2. Aprendizagem Federada

Aprendizagem Federada é uma técnica de ML que tem como uma das suas principais características a distribuição dos dados e do seu processamento que depois serão agregados em um modelo de aprendizagem final.

A ideia de FL é que ao invés de concentrar dados e processamento para aprendizagem em um apenas um local,como ocorre nos modelos de aprendizagem de máquina mais comuns. Os dados são processados localmente na instância chamada de cliente, nela os dados são adquiridos/armazenados e apenas o modelo local resultante é enviado para um servidor central que agrega os modelos em um modelo global. Então esse modelo global é repassado para os clientes que auxiliam no treinamento do modelo do cliente.

Para ser considerado uma aprendizagem federada, o modelo encontrado pelos clientes deve ser agregado de alguma forma ao modelo do servidor e repassado o modelo agregado para os demais clientes. O modelo de agregação mais conhecido é o “Federated Averaging”(ou FedAvg), que simplesmente gera o modelo global realizando a média dos modelos recebidos dos clientes antes de repassar os mesmos para todos os clientes novamente.

Segundo B. Yurdem(2024), as principais características de uma FL são :

- Amostragem enorme de dados: Possivelmente usa todos os dados disponíveis da instituição como base de treinamento.
- Amostragem diversa de dados “non-IID” de diferentes dispositivos: Muitas vezes os dados são não identicamente distribuídos nos nodos de amostragem de dados e nodos podem ter mínimos ou até nenhum dado.
- Descentralizado: a diferença entre clientes e servidores é organizacional, mesmo que cada cliente não seja o centro, ele é capaz de influenciar o modelo global.
- Igualdade de nodos de uma FL: Todas as partes de uma FL possuem os mesmos parâmetros, o que diferencia os clientes entre si são a quantidade de dados que cada um tem acesso

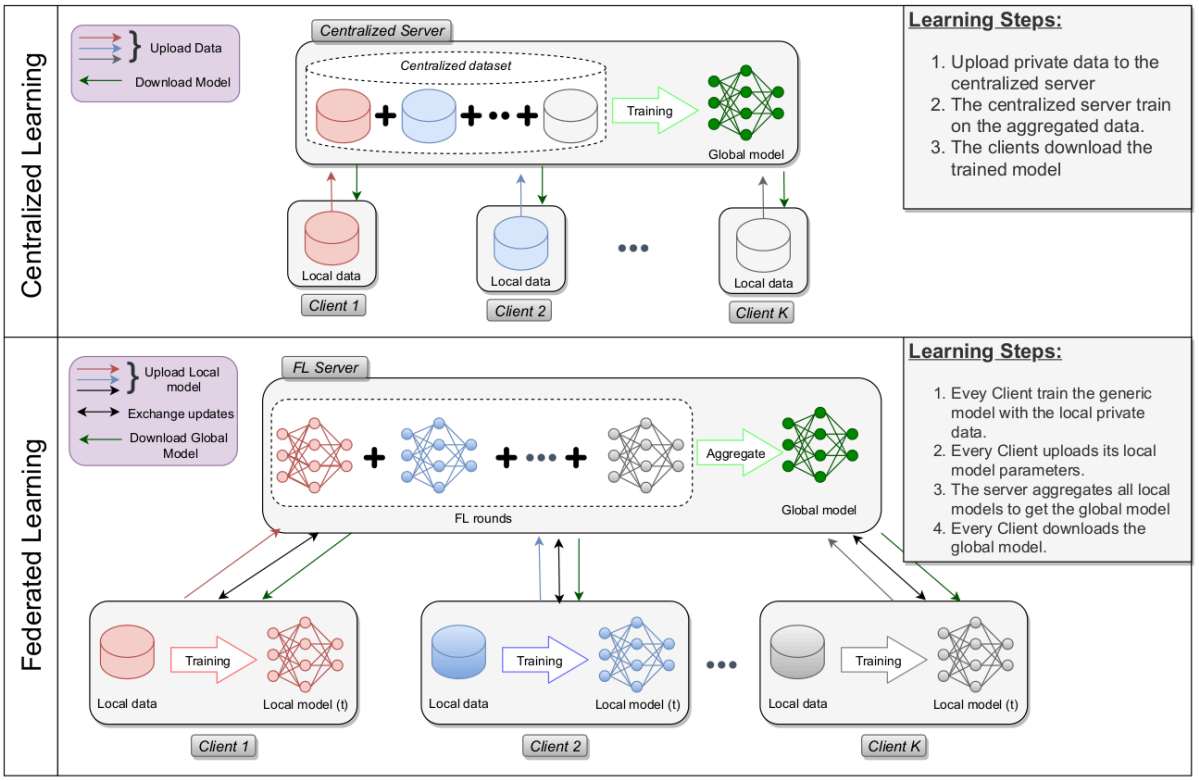


Fig. 2.1 : Comparação entre Aprendizagem Centralizada e FL (M. A. Ferrag, 2021)

A figura acima(Fig. 2.1) mostra o esquema de comunicação e aprendizado realizado por FL e como ele se difere de um modelo de aprendizado centralizado. Note que o modelo Centralizado recebe dados de diversos clientes diferentes e mantém esses dados em um servidor centralizado antes de realizar a aprendizagem desses dados localmente e gerar um modelo global no servidor que é repassado aos clientes. Já no esquema do FL, cada cliente gera os seus próprios modelos com os dados disponíveis localmente, antes de repassar os modelos do cliente para o servidor que realiza a agregação para gerar o modelo global. Em seguida, o modelo global é repassado para todos os clientes. Assim, a cada ciclo de aprendizado a única comunicação entre cliente e servidor são os modelos criados a cada iteração de aprendizagem.

Ou seja, as seguintes fases estão presentes no treinamento de uma FL:

- Seleção de cliente: clientes que participaram do treinamento são selecionados.
- Comunicação: Clientes recebem o modelo global para usar como base durante o treinamento.
- Computação do cliente: Cada cliente treina um modelo local usando os dados localmente disponíveis.
- Agregação: O servidor recebe os modelos locais dos clientes e os agrega usando uma das técnicas de agregação.

- Atualização do modelo: Modelo global é atualizado.

2.2.1 Vantagens e Limitações

Essa técnica traz algumas vantagens sobre a técnica tradicional de aprendizagem centralizada, como por exemplo, evitar o transporte de dados entre cliente e servidor, evitando tráfego na rede e riscos de vazamento de dados privados. Outra vantagem é que o processamento de dados é distribuído, assim os recursos disponíveis são melhor aproveitados e não há tanto estresse no servidor.

Uma das aplicações positivas é que o recolhimento e armazenamento dos dados pode ser realizado pelos dispositivos dos processos clientes, já que eles estão fisicamente próximos de onde os dados reais se encontram, enquanto o servidor não sofre pela transferência excessiva de dados coletados, pois apenas o modelo após o aprendizado é transferido.

Apesar de todas as vantagens do FL sobre a aprendizagem centralizada, o FL é suscetível a ataques maliciosos, como ataques “poisoning” que buscam se passar por um cliente e modificar o modelo local para comprometer a acurácia do modelo global ou ataques ou ataques “jamming” que buscam interromper a comunicação entre cliente e servidor.

Por outro lado, a própria arquitetura do FL pode causar falhas pela diferença dos dados que cada cliente tem acesso. Por exemplo, dados utilizados entre clientes podem ser “Non-IID”, isso significa que os dados não são independentes entre si. Outra possibilidade é que os dados podem estar desbalanceados, situação a qual clientes possuem muita representação de um tipo de classificação ou pouca representação de outro tipo.

Outro problema menos óbvio é a limitação de hardware e de comunicação entre clientes e servidores. Hardware pode resultar em comunicação lenta ou instável, assim como um ambiente desfavorável, como cidades que possuem diversos sinais de diferentes dispositivos e podem realizar interferência nas comunicações entre servidor e cliente. Dessa forma, características do ambiente de implementação podem prejudicar negativamente o aprendizado da rede como um todo.

De forma similar, os clientes de uma FL podem possuir especificações diferentes entre si, na qual armazenamento e processamento de hardware são limitadas relativamente entre si. Isso pode causar situações nas quais a eficiência de treinamento do modelo pode ser afetada negativamente.

2.2.2 Tipos de Aprendizagem Federada

Como as características do ambiente e dos dados afetam significativamente os clientes e o aprendizado da rede como um todo, é necessário saber que tipo de FL se deseja implementar para a utilização efetiva do modelo. Os três modelos a

seguir são possíveis de tratamento de dados: aprendizagem federada horizontal, aprendizagem federada vertical e aprendizagem federada de transferência. Um esquema simples de como cada um funciona se encontra na imagem(Fig. 2.2) a seguir:

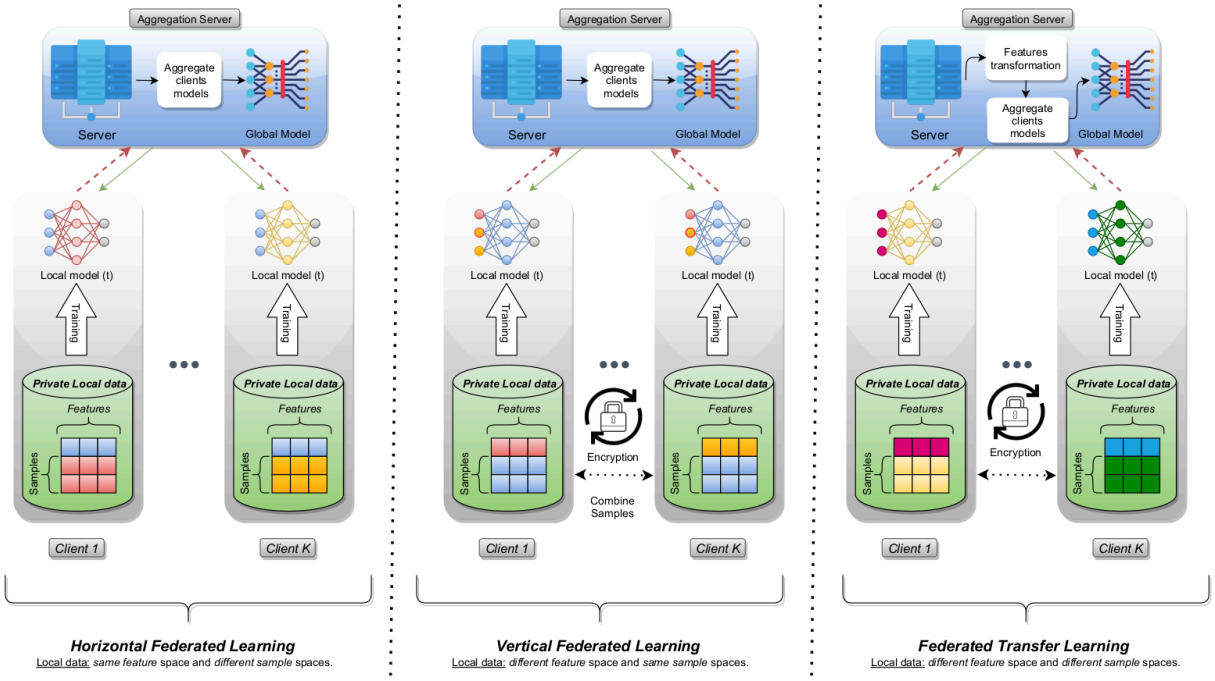


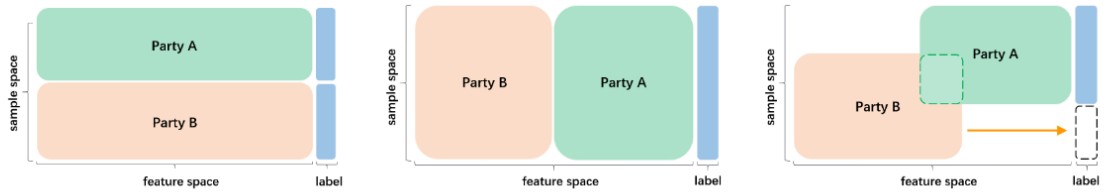
Fig. 2.2 : Os diferentes tipos de *Federated Learning* (M. A. Ferrag, 2021)

Aprendizagem federada horizontal é utilizada quando os clientes possuem dados com as mesmas características porém espaço de amostragem diferente (survey), esse é o modelo que é utilizado neste trabalho e o mais comum por ser simples de ser implementado. Por exemplo, Google usa FL horizontal para permitir usuários de dispositivos celulares usarem os seus dados e colaborarem para o treinamento de um modelo de “next-word prediction”(H. B. McMahan,2016).

Aprendizagem federada vertical é utilizada quando os clientes possuem a mesma amostragem, porém cada cliente possui um subgrupo incompleto das diferentes características dessa amostragem. Por exemplo, Webank usa FL vertical para construir modelos de risco para empresas que contratam seus serviços(Y. Cheng,2020)

Aprendizagem federada de transferência é utilizada quando a amostragem e as suas características são diferentes entre os clientes, na qual pode haver ou não coincidência de amostragem ou características.Por exemplo, dados de Eletroencefalografia (sensores de sinais elétricos no cérebro) de múltiplos pacientes

de regiões cerebrais diferentes são usadas juntamente com FL de transferência para montar um modelos de Interface cérebro-computador(C. Ju, 2020).



(a) Horizontal Federated Learning (b) Vertical Federated Learning (c) Federated Transfer Learning

Fig. 2.3 : Como os diferentes tipos de FL tratam os dados (Y. Liu, 2023)

Na figura acima (Fig.2.3), temos a exemplificação de como são as características dos dados de diferentes FL. Na figura (a) temos a FL horizontal, que possui as mesmas características(em azul)entre amostragens, porém as se diferenciam entre amostras(verde e amarelo). Na figura (b) temos a FL vertical, que possui diferentes características(azul), porém a amostragem é a mesma(em verde e amarelo). Por fim, temos a FL de transferência que possui ambas características e amostragens diferentes. Note que, na imagem há uma intersecção entre as diferentes amostragens e também há intersecção entre diferentes características, porém não há garantias que isso ocorra nas bases de dados da FL de transferência.

3. Aplicações de Aprendizagem Federada

O FL pode ser usado em diversos ambientes em que suas características são desejadas como em redes IoT, tecnologias da blockchain, segurança de redes, processamento de dados, entre outras. Nesta parte do trabalho será apresentado alguns dos usos da FL em soluções do mundo real.

3.1 Aprendizagem Federada em segurança

O uso de câmeras para sistemas de segurança é uma ideia altamente difundida. Mas com o uso de FL e drones um sistema pode ser criado para usar reconhecimento facial e identificar possíveis indivíduos suspeitos(N. H. Motlagh, 2017) . As vantagens ao combinar essas duas ideias são um sistema com processamento mais eficiente e tempo de reconhecimento facial reduzido.

Drones podem ser usados como câmeras móveis para realizar a segurança ou inspeção de um local, porém eles também podem ser alvos de ataques ou serem usados por agentes com más intenções, como gravar fotos e vídeos sem ser descoberto. Para melhor controle de quais drones são permitidos voos e em quais lugares podemos usar uma combinação de FL e frequências de rádio presentes em redes IoT(A. Yazdinejad, 2021). Dessa forma a autenticação é feita localmente nos drones, enquanto combate problemas de escalabilidade, privacidade e segurança de dados.

A necessidade de processar muitos dados e manter o sigilo dos mesmos pode ser um problema, por exemplo, processar os dados remotamente com computação em nuvem é uma opção interessante para quem não pode ou não quer manter uma infraestrutura fisicamente próxima, porém surge a preocupação com segurança desses dados. Mas ao utilizar FL e um protocolo específico de privacidade e comunicação é possível reduzir o tempo de execução desse processamento em 20% e tamanho de mensagens criptografadas que carregam esses dados em uma média de 85% (C. Fang, 2020) sem a perda de segurança fornecida por outros métodos.

3.2 Aprendizagem Federada em IoT

Dispositivos IoT têm se tornado cada vez mais comuns no nosso cotidiano e muitos são vulneráveis por conta do design, implementação ou configuração. Infelizmente, técnicas atuais podem não ser boas para detectar dispositivos que já estão comprometidos, mas com o uso de FL dentro do esquema DÏoT (T. D. Nguyen, 2019) dispositivos comprometidos podem ser detectados com 95.6% das vezes , aumentando a segurança em casas com dispositivos inteligentes.

Outra característica comum desses dispositivos IoT é que é comum para dados serem transferidos regularmente entre diferentes domínios de segurança de diferentes provedores de serviço causando que centros de dados fiquem expostos a possíveis falhas de segurança. Mas com o uso de FL combinado com tecnologia de aprendizado profundo é possível criar um framework que não remove a necessidade de transferência de dados entre diferentes centros de dados privados(B. Yin, 2020).

O uso de IoT tem se tornado cada vez mais comum em indústrias e instituições de pesquisa, que por sua vez têm explorado a possibilidade do uso de 6G e suas vantagens em comunicação. Felizmente, muitos fatores limitantes podem ser tratados com o uso de FL, lidando com preocupações de privacidade, processamento centralizado em aplicações IoT (Y. Liu et al.,2020).

O uso de IoT e FDL em veículos pode fornecer uma experiência mais segura e confortável, abrindo um novo caminho para carros autônomos se destacarem, no qual locais com problemas de recepção e estabilidade de GPS podem ainda funcionar com privacidade e flexibilidade dos demais carros (Q. Kong, 2021). Dessa forma

3.2 Aprendizagem Federada em processamento de dados

Ao combinar as características de descentralização de FL e a segurança de blockchain existe a possibilidade de combinar em uma “Mobile Edge Computing” que processa todos os dados em uma rede descentralizada sem a necessidade de um servidor. Esse novo paradigma foi batizado de FLchain (D. C. Nguyen, 2021) e ele permite o uso de dispositivos móveis para o processamento de dados sem a preocupação de privacidade, podendo ser usado em várias soluções envolvendo “edge data sharing”, “edge content caching” e “edge crowdsensing”.

FL também pode ser usado em um contexto de saúde para permitir o treinamento de uma rede de dados de pacientes sem a preocupação de privacidade dos pacientes. Com o crescimento de novos tipos de doenças e sintomas cada vez mais complexos, o conhecimento é uma das armas mais importantes para manter o gerenciamento efetivo do sistema de saúde. Assim, a capacidade de treinar modelos em bases significativas de pacientes sem se preocupar com o sigilo pode criar uma ferramenta (C. M. Thwal, 2024) que ajuda equipes médicas a realizar diagnósticos.

Sistemas ciber-físicos-sociais combinam sistemas de comunicação, físicos e computacionais para atingir novas capacidades em coletar e armazenar dados com informação privada ou informações incompletas armazenados em localizações diferentes. Esses sistemas possuem dificuldades de escalabilidade, dispersão e

segurança, mas que podem ser melhorados com o uso de FL(J. Yang,2021). Ao utilizar FL e algoritmos corretos é possível aliviar essas dificuldades e economizar tempo e espaço .

4. Uso de Aprendizagem Federada

Para exemplificação deste trabalho será usado o Framework Flower (FF) e o projeto básico fornecido como tutorial está disponível no github dos fundadores da FF, que utiliza a base de dados CIFAR-10.

Essa base em questão é famosa pelo treinamento de redes neurais em reconhecimento de imagens. Ela possui a distribuição uniforme de 10 classes em 60 mil imagens de 32x32 pixels com três canais RGB. Algumas das classes dessas imagens são animais (gatos, cachorros, pássaros, etc) enquanto outras são meios de locomoção(avião, barco, caminhão, etc).

Ao iniciar o projeto nos deparamos com a estrutura de diretórios com os arquivos que serão utilizados durante os testes :

- _init_.py_ : arquivo de inicialização do python.
- client_app.py : arquivo com as definições e configurações do Cliente da FL.
- server_app.py : arquivo com as definições e configurações do Servidor da FL.
- task.py : Define o modelo, treinamento e processo da base de dados que serão utilizados durante a FL.
- pyproject.toml : Arquivo com os metadados de controle do projeto, como configurações e controle de versão.

```
quickstart-pytorch
├── pytorchexample
│   ├── __init__.py
│   ├── client_app.py   # Defines your ClientApp
│   ├── server_app.py   # Defines your ServerApp
│   └── task.py         # Defines your model, training and data loading
├── pyproject.toml      # Project metadata like dependencies and configs
└── README.md
```

Fig. 4.1 : Diretório de arquivos do projeto básico

Como os arquivos client_app.py e server_app.py apresentam em maior parte as configurações do cliente e do servidor, respectivamente, ambas que podem ser reconfiguradas no arquivo pyproject.toml. Logo, não será dado tanto foco para esses arquivos, nós focaremos em task.py que define o modelo de e dados utilizados e em pyproject.toml que define o comportamento da FL.

4.1 Explicação da CNN

O modelo de CNN que será usado está definido no arquivo task.py e é um modelo relativamente simples, que possui 6 camadas no total, essas divididas em de convolução, pooling e lineares. Como está na imagem (Fig. 4.2) a seguir:

```python
class Net(nn.Module):
    """Model (simple CNN adapted from 'PyTorch: A 60 Minute Blitz')"""

    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 16 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        return self.fc3(x)
```

Fig. 4.2 : Estrutura da CNN presente em task.py

A primeira e a terceira camada são inicializadas com três números. O primeiro deles representa o número de canais de entrada nessa camada, enquanto o segundo representa o número de canais que saem dessa camada. Finalmente, o terceiro número representa o tamanho do kernel que será usado dentro dessa camada, Como é um único número inteiro, essa será a altura e profundidade dos nodos presentes nessa camada, ou seja, é um kernel de 5x5 em ambas camadas.

A segunda camada é uma camada de pooling, o primeiro parâmetro representa um kernel de tamanho 2x2 que agrega o valor máximo de cada 4 dados em um novo dado para próxima camada, o segundo parâmetro representa o passo que esse kernel realizará antes de ser aplicado em novos dados. Um passo do mesmo tamanho que o kernel representa que o kernel não irá manipular o mesmo dado mais de uma vez, pois o passo é grande o bastante para evitar intersecção de dados.

As últimas três camadas são chamadas de camadas de convolução(essas camadas são chamadas de lineares pelo programa, porém nesse exemplo, elas são sinônimos), nas quais o primeiro parâmetro é o tamanho dos dados de entrada e o segundo é o tamanho dos dados de saída. Note que essas camadas estão ligadas seriamente, então a saída da camada anterior se conecta diretamente com a entrada da próxima camada até chegar na camada final que as 10 saídas coincidem com as 10 possíveis classificações de imagem dos dados que entraram na rede.

4.2 Ambiente da Aprendizagem Federada

No arquivo pyproject.toml, temos as configurações gerais de como a FL funciona, nesse arquivo também é possível sobrescrever algumas das configurações

dos arquivos de cliente e de servidor. Valores como taxa de aprendizado, tamanho de *batch*, entre outros.

Como o foco deste trabalho é o FL, vamos apresentar com mais cuidado as características mais relevantes ao FL e ao seu uso. As características específicas da CNN não são de muita importância, então será entrado em detalhes. No arquivo pyproject.toml, temos as seguintes configurações do FF:

- Iterações do Servidor: Define quantas vezes haverá a comunicação entre servidor-cliente e quantos modelos globais serão gerados, já que cada iteração do servidor acaba com o modelo global gerado sendo enviado para os clientes. Por padrão, definiremos como 100.
- Épocas locais: Define quantas iterações cada cliente fará com seu conjunto de dados antes de enviar o seu modelo ao servidor. Por padrão, definiremos como 1.
- Taxa de aprendizado:Define o quão rápido o modelo aprende e se adapta a novos dados. Por padrão, definiremos como 0,1.
- Tamanho de *Batch*: Tamanho do conjunto de dados processados durante uma iteração local do cliente antes de se atualizar os parâmetros do modelo do cliente. Por padrão, definiremos como 32.
- Número de clientes: Quantidade de clientes que participaram do teste. Por padrão, definimos como 10.
- Fração de avaliação: Essa configuração define qual é a porcentagem de clientes que será selecionada pelo servidor para receber uma avaliação. Nesse caso faremos apenas uma avaliação simples de acurácia. Por padrão, definiremos como 0,5.

Outras características do teste não inclusas no arquivo de configuraçẽos gerais são o modelo de agregação que é o FedAvg, e as partições são criadas com um particionador que seleciona um subconjunto de dados aleatórios do banco de dados CIFAR-10. Para separar os dados usado durante a aprendizagem dos clientes, foi separada 20% da base CIFAR-10 com amostragem aleatória e dividido entre os clientes. Além disso, a mesma *seed*(elemento que controla a aleatoriedade de amostragem nos testes) foi utilizada para realizar essa separação de amostras para os clientes, assim garantindo que todos os testes ocorreram com as mesmas condições de amostragem do banco de dados.

4.3 Testes Realizados

Os testes realizados foram feitos em um computador sem outros programas rodando simultaneamente e com internet desconectada para evitar troca de pacotes no background do sistema operacional durante os testes. A seguir estão as informações do computador:

- Sistema Operacional: Ubuntu 22.04.5 LTS

- Processador: Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz com 8 CPUs
- Caches: L1d de 192KiB, L1 de 128KiB, L2 de 2MiB e L3 de 8 MiB

Como o objetivo deste teste é avaliar como FL se comporta em diferentes configuraçẽos, as únicas opções que alteramos serão as de FL. De forma similar utilizaremos os ciclos de iteração do servidor e as suas respectivas acurácias para avaliar como cada parâmetro afetou o resultado. Neste trabalho foi escolhido não avaliar o tempo de cada teste em relação aos demais parâmetros, isso se deve ao uso de FL e os seus respectivos clientes serem muito dependentes ao hardware e ambiente que se encontram, além da possibilidade da comunicação entre eles falharem. Assim, não usaremos tempo como métrica para não causar complicações desnecessárias tentando simular o tempo de uma aplicação real que pode ou não estar correta.

Nesse gráfico(Fig. 4.3) a seguir temos o resultado dos testes que variam as frações de avaliação, estamos variando nossos resultados entre 10% e 100% do uso dos clientes disponíveis para realizar a avaliação do servidor . Podemos notar que não importa a fração de avaliação escolhida, os resultados são todos similares, uma possível explicação para esse comportamento é que por conta da base de dados CIFAR-10 ser uma base de dados bem distribuída. Dessa forma, mesmo com as partições aleatórias, os clientes possuem uma amostragem grande e balanceada o bastante para que cada cliente representa satisfatoriamente o aprendizado que o servidor deseja incluir no modelo global.

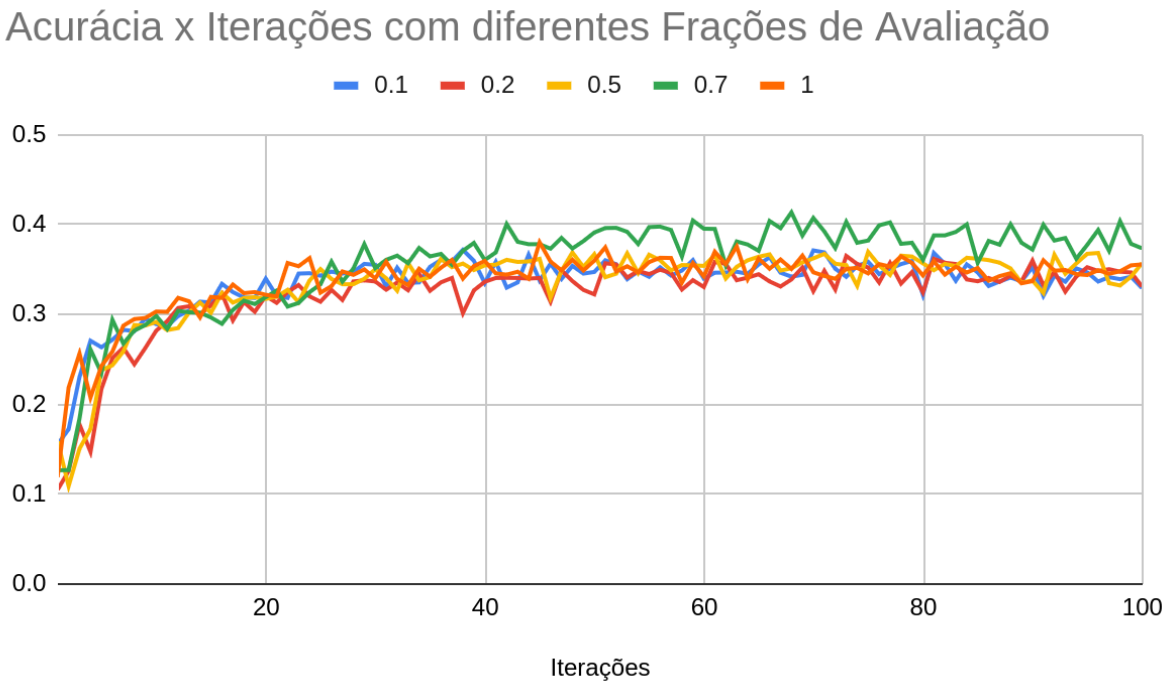


Fig. 4.3 : Gráfico de como diferentes frações de avaliação afetam a acurácia

No próximo gráfico(Fig. 4.4) temos como as taxas de aprendizado afetam a acurácia dos nossos testes. A taxa de aprendizado varia entre 0.1 e 0.4, porém o melhor resultado está com o valor de 0.1 enquanto os demais rapidamente perdem a acurácia. Note que assim como aprendizagem centralizada, uma taxa de aprendizagem maior entre vários clientes diferentes não melhora os resultados significativamente mais que utilizar um valor menor que progressivamente tem melhora na acurácia. Ainda mais, o modelo FL também tem uma queda brusca após certo número de iterações, isso também pode ser explicado pela condição de overfitting do modelo FL, essa condição problemática também ocorre nos modelos centralizados. Existem vários jeitos de combater o overfitting, o mais simples é pausar a aprendizagem quando a acurácia cair e utilizar apenas o modelo mais eficiente encontrado. Outras soluções possíveis são aumentar a base de dados utilizada para o treinamento ou até mesmo simplificar o modelo utilizado para aprendizagem.

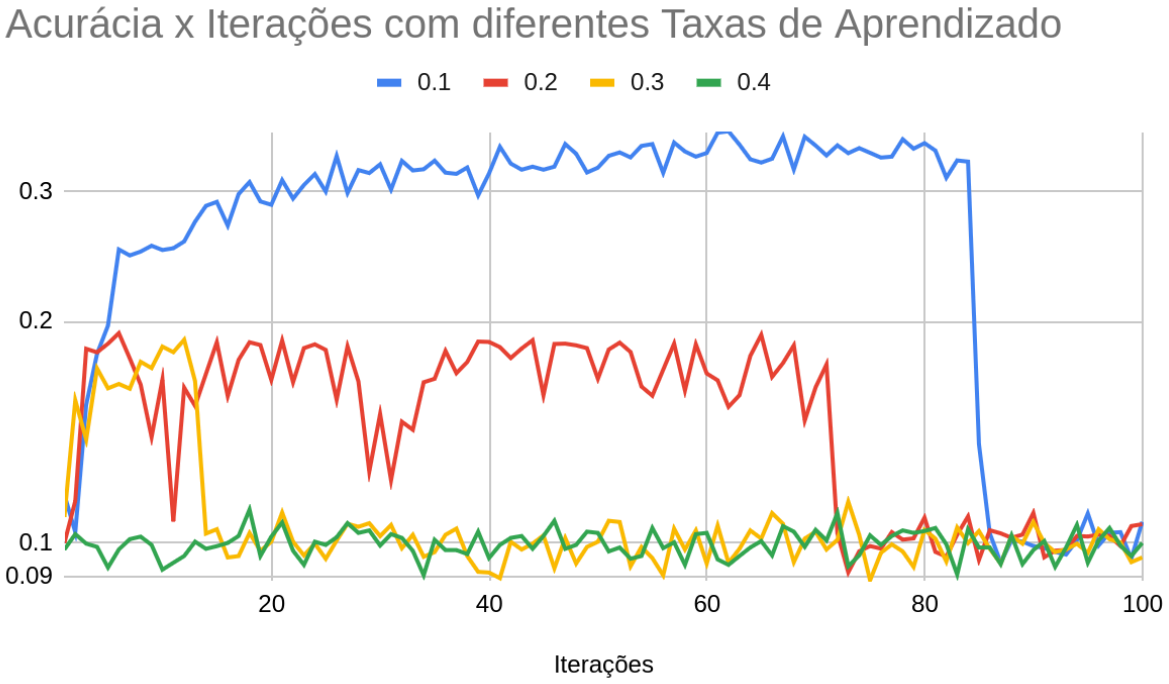


Fig. 4.4 : Gráfico de como diferentes taxas de aprendizado afetam a acurácia

Por fim temos o último gráfico(Fig. 4.5) que varia a quantidade de épocas locais que o cliente realiza antes de fazer a comunicação com o servidor e enviar o modelo local, no qual apresenta valores que variam entre 1 e 21. Note que a melhor acurácia veio do teste com apenas uma época local e que os demais testes foram similarmente piores entre si. Uma possível explicação para isso provavelmente é o alto número do *batch* (que é 32 como mencionado anteriormente) e que épocas extras não fazem que o cliente consiga extrair mais dados nem padrões significativos da amostragem que ele recebe. Pelo contrário, um alto número de *batchs* piora a acurácia dos testes realizados. Além disso, a baixa acurácia de

épocas maiores também pode ser resultado de overfitting que acaba prejudicando negativamente a acurácia.

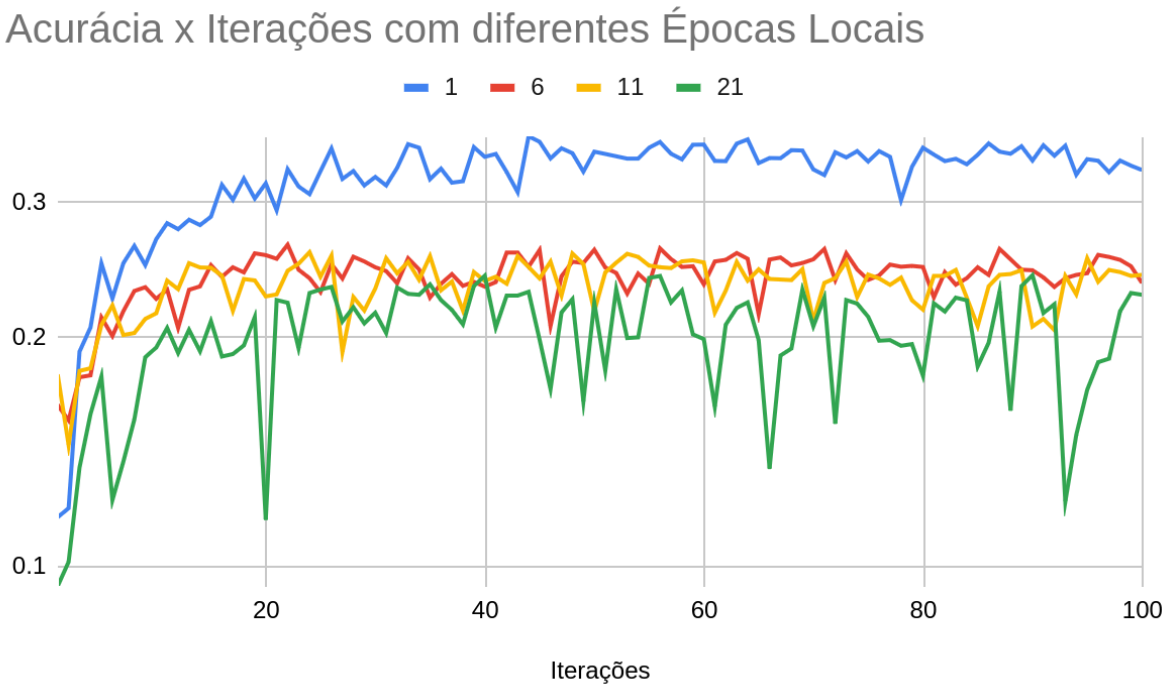


Fig. 4.5 : Gráfico de como diferentes épocas locais afetam a acurácia

5. Conclusão

Este trabalho explorou conceitos de ML, como perceptrons e organização RN em camadas. Também foram apresentados conceitos de uma CNN e como diferentes camadas se comportam e como elas se conectam entre si.

Em seguida, conceitos de FL foram apresentados, como os diferentes tipos de FL e como a comunicação entre cliente e servidor ocorre e quais informações são transmitidas entre eles. Além disso, o código exemplo de uma FedAvg foi apresentado com a utilização de FF e explicação das partes mais importantes do código de uma FL funcional.

Ao longo do trabalho também foi explorado possíveis usos da FL para implementações de sistemas de segurança, dispositivos IoT e processamento não centralizado. Essas implementações variam desde da área de saúde, uso doméstico e até de grandes empresas. Isso mostra que as características disponibilizadas por uma FL são abrangentes e desejadas em vários setores diferentes.

Para trabalhos futuros, uma das possibilidades é explorar modelos de FL que utilizam técnicas diferentes para agregar modelos de servidor e clientes e como essas técnicas afetam a velocidade de aprendizagem. Por exemplo, FL com gradientes descendentes (“Federated stochastic gradient descent” ou FedSGD) agregam os modelos de cliente e servidor baseados na quantidade de amostragem que cada cliente teve acesso naquela iteração específica, assim clientes com maior amostragem se tornam mais importantes que clientes com menos dados, assim, podendo atingir uma convergência de modelo mais preciso com menos iterações.

Outra possibilidade de trabalhos futuros é explorar como diferentes bases de dados se comportam em uma FL, e como uma base de dados desbalanceada afeta o FL ou com quantas classificações diferentes os clientes de uma FL conseguem manter antes do processo de agregação de modelos começar a ser afetada negativamente e ocorrer perda de acurácia.

Referências

A. Yazdinejad, R. M. Parizi, A. Dehghantanha, H. Karimipour,”Federated learning for drone authentication” em “Ad Hoc Networks”, Volume 120, 2021

B. Yin, H. Yin, Y. Wu e Z. Jiang, "FDC: A Secure Federated Deep Learning Mechanism for Data Collaborations in the Internet of Things," em “*IEEE Internet of Things Journal”*, vol. 7, páginas 6348-6359, 2020

B. Yurdem, M. Kuzlu. “Federated learning: Overview, strategies, applications, tools and future directions” em “Heliyon”, 10(19), 2024

C. Fang, Y. Guo, N. Wang, e A. Ju, “Highly efficient federated learning with strong privacy preservation in cloud computing”, em “Computers & Security”, vol. 96, página 101889, 2020.

C. Ju, D. Gao, R. Mane, et al., “Federated transfer learning for eeg signal classification”, em “IEEE Engineering in Medicine and Biology Society”, 2020.

C. M. Thwal, K. Thar, Y. L. Tun, e C. S. Hong.“Attention on personalized clinical decision support system: Federated learning approach”, em “2021 IEEE International Conference on Big Data and Smart Computing”. IEEE, páginas. 141–147, 2021.

D. C. Nguyen, M. Ding, Q.-V. Pham et al., “Federated learning meets blockchain in edge computing: Opportunities and challenges,” em “IEEE Internet of Things Journal”, 2021.

D. Man, F. Zeng, W. Yang et al., “Intelligent Intrusion Detection Based on Federated Learning for Edge-Assisted Internet of Things” em “Security and Communication Networks”, Volume 2021, Primeira edição.

E. Fedorchenko, E. Novikova e A. Shulepov, “Comparative Review of the Intrusion Detection Systems Based on Federated Learning: Advantages and Open Challenges” em “Algorithms 2022”, 15(7), 247

E. Novikova, E Doynikova e S. Golubev, “Federated Learning for Intrusion Detection in the Critical Infrastructures: Vertically Partitioned Data Use Case” em “Algorithms 2022”, 15(4), 104

H. B. McMahan, E. Moore, D. Ramage, e B. A. y Arcas.“Federated learning of deep networks using model averaging”, em CoRR, vol. abs/1602.05629, 2016.

H. B. McMahan, E. Moore, D. Ramage et al., “Communication-Efficient Learning of Deep Networks from Decentralized Data” em” JMLR: W&CP”, volume 54, 2017

I. Mohammed et al., "Budgeted Online Selection of Candidate IoT Clients to Participate in Federated Learning," em “IEEE Internet of Things Journal”, vol. 8, no. 7, pp. 5938-5952, 1 Abril, 2021

J. Yang, C. Fu, H. Lu, “Optimized and federated soft-impute for privacy-preserving tensor completion in cyber-physical-social systems” em “Information Sciences”, vol. 564, páginas 103-123, 2021

J. Yoon, W. Jeong and G. Lee et al., “Federated Continual Learning with Weighted Inter-client Transfer” em ICML 2021.

M. A. Ferrag, O. Friha, L. Maglaras, H. Janicke e L. Shu, "Federated Deep Learning for Cyber Security in the Internet of Things: Concepts, Applications, and Experimental Analysis," em IEEE Access, vol. 9, página 138509-138542, 2021

N. H. Motlagh, M. Bagaa, e T. Taleb, “Uav-based iot platform: A crowd surveillance use case”, em “IEEE Communications Magazine”, vol. 55, páginas 128–134, 2017.

PAN, Hen.”Federated Learning with PyTorch and Flower (Quickstart Example) ”. **Github,** 2024. Disponível em: https://github.com/adap/flower/tree/main/examples/quickstart-pytorch. Acesso em: 04, outubro,2024

Q. Kong, F. Yin, R. Lu, et al.,"Privacy-Preserving Aggregation for Federated Learning-Based Navigation in Vehicular Fog," em “*IEEE Transactions on Industrial Informatics”*, vol. 17, no. 12, páginas 8453-8463, 2021

S. Russell, P. Norvig , “Inteligência artificial” , tradução R. C. Simille, Rio de Janeiro: Elsevier, 2013.

T. D. Nguyen, S. Marchal, M. Miettinen et al., "DÏoT: A Federated Self-learning Anomaly Detection System for IoT," em “*2019 IEEE 39th International Conference on Distributed Computing Systems”*, páginas 756-767,2019

W.S McCulloch, W. Pitts. “A logical calculus of the ideas immanent in nervous activity” em “Bulletin of Mathematical Biophysics 5”, páginas 115–133, 1943

Y. Cheng, Y. Liu, T. Chen, e Q. Yang. “Federated learning for privacy-preserving ai”, em Commun. ACM, volume 63, páginas 33–36, 2020

Y. Liu, Y. Kang, T. Zou et al., “Vertical Federated Learning: Concepts, Advances and Challenges” em ”IEEE Transactions on Knowledge and Data Engineering”, 2024

Y. Liu, X.Yuan, Z. Xiong, et al.,“Federated learning for 6g communications: Challenges, methods, and future directions”, em “China Communications”, vol. 17, no. 9, páginas. 105–118, 2020.