Análise do Aprendizado Federado em Redes Móveis

Kaylani Bochie[1], **Matteo Sammarco**[2],
Marcin Detyniecki[2] **e Miguel Elias M. Campista**[1]

[1]Grupo de Teleinformática e Automação (GTA)
PEE/COPPE-DEL/Poli
Universidade Federal do Rio de Janeiro (UFRJ)
[2]AXA

{kaylani,miguel}@gta.ufrj.br,
{matteo.sammarco,marcin.detyniecki}@axa.com

Resumo. *A aquisição cada vez maior de informações pessoais para personalização de serviços coloca em cheque a privacidade dos usuários. No contexto do aprendizado federado, a privacidade pode ser preservada com o compartilhamento apenas de pesos sinápticos de redes neurais entre clientes e servidores. Este trabalho avalia o impacto de diferentes parâmetros de redes de computadores no desempenho de modelos de aprendizado federado em um cenário composto por clientes móveis. Para isso, o desempenho das redes neurais convolucionais para classificação de imagens é considerado com o uso do conjunto de dados CIFAR-10. Os experimentos realizados utilizam o framework* `Flower` *para avaliar parâmetros comuns em redes móveis como latência, conectividade, volume de dados e disponibilidade dos clientes. Os resultados indicam que, além do aumento no tempo total de treinamento, um aumento no número de usuários desconectados em uma rodada de treinamento pode até mesmo reduzir o desempenho do modelo federado. Esses resultados reforçam a necessidade de orquestração cliente-servidor para adaptação dinâmica às condições de rede.*

Abstract. *The increase of personal data collection for service customization threatens users' privacy. In the context of federated learning, privacy can be preserved by distributing the learning process, where only neural networks' weights are shared between clients and servers. This work evaluates the impact of different computer networks' parameters on the performance of federated learning models in a mobile oriented scenario. For this, we consider the performance of convolutional neural networks for image classification. The experiments carried out use the* `Flower` *framework and the CIFAR-10 dataset for image classification. Common parameters in mobile networks such as latency, connectivity, data volume, and client availability are evaluated. The results indicate that, in addition to the increase in total training time, an increase in the number of disconnected users in a training round can even reduce the performance of the federated model. These results indicate the need for client-server orchestration to perform dynamic adaptation of network conditions.*

1. Introdução

A popularização de soluções baseadas em aprendizado profundo em redes de computadores é inegável. Frações cada vez maiores da literatura se dedicam à criação

e à avaliação de técnicas que possam melhorar o desempenho das redes de computadores, maximizando assim a Qualidade de Experiência (*Quality of Experience* – QoE) dos usuários finais [Bochie et al., 2020]. Entretanto, nos últimos anos, a privacidade dos dados utilizados para o treinamento de modelos de aprendizado profundo vem estimulando novas áreas de pesquisa [Shokri e Shmatikov, 2015]. Esforços são feitos em temas como Criptografia Homomórfica (*Homomorphic Encryption*), onde certas operações podem ser executadas nos dados encriptados sem a necessidade de decifrá-los [Hernandez Marcano et al., 2019], ou privacidade diferencial, que busca introduzir ruído aos dados transmitidos para inviabilizar a identificação dos produtores de dados [Dwork e Roth, 2014]. Porém, mesmo utilizando dados encriptados ou anonimizados, ainda existe a necessidade de concentrar os dados em um servidor para o treinamento, em cenários de aprendizado de máquina centralizados. Assim, os riscos de vazamento e identificação de padrões persistem, sendo possíveis até mesmo a partir de características demográficas dos dados coletados [Nguyen, 2011, Narayanan e Shmatikov, 2008].

O Aprendizado Federado (*Federated Learning*) surge da necessidade de evitar a transferência dos dados coletados por cada dispositivo para a nuvem, mantendo assim a privacidade dos usuários, porém ainda possibilitando soluções baseadas em aprendizado profundo [McMahan et al., 2017]. O aprendizado federado se baseia na distribuição do treinamento de modelos de aprendizado, normalmente Redes Neurais Profundas (*Deep Neural Networks* – DNNs), nos dispositivos de borda. Cada dispositivo treina uma DNN em seu conjunto de dados privado e reporta os parâmetros da DNN para um servidor central. Este servidor é responsável pela agregação dos parâmetros em um modelo completo e pela distribuição do modelo agregado para os dispositivos de borda realizarem inferências. Este desacoplamento entre trabalho computacional e aquisição de dados garante a privacidade dos dispositivos participantes e reduz o custo de comunicação, visto que os parâmetros transmitidos são menores que o conjunto de dados [Cunha Neto et al., 2020]. Ademais, a capacidade de atualizar os modelos iterativamente para realizar inferências garante também maior eficiência no uso dos enlaces e redução de latência durante a etapa de inferência [Lim et al., 2020].

Apesar das vantagens do aprendizado federado, este vem acompanhado também de desafios. As principais características e os consequentes desafios do aprendizado federado em relação a cenários de aprendizado de máquina centralizado podem ser resumidos em quatro. Primeiro, a presença de dados não IID (Independentes e Identicamente Distribuídos), ou seja, o padrão de uso de cada usuário é refletido na composição do conjunto de dados local, o que por sua vez faz com que o modelo local se especialize em uma distribuição de dados. Segundo, o volume de dados é não uniforme. Como diferentes clientes no cenário federado geram volumes de dados diferentes, cada conjunto de parâmetros reportado ao servidor central pode ser mais ou menos representativo da distribuição de dados global. Terceiro, o número de participantes envolvidos no aprendizado é variável. Logo, o servidor deve possuir mecanismos para lidar com conexões intermitentes e com a inclusão de novos clientes durante uma rodada de treinamento. Por último, as limitações de *hardware* dos participantes são heterogêneas. A arquitetura de rede neural escolhida deve ser idêntica para que o servidor possa agregar os parâmetros de cada cliente. Porém, dispositivos limitados podem não ser capazes de contribuir para o treinamento de modelos muito complexos, enquanto dispositivos com maior poder computacional podem acabar sendo subutilizados ao escolher modelos de

aprendizado muito simples. Consequentemente, é necessário escolher a arquitetura de rede neural apropriada para otimizar globalmente o desempenho em todos os dispositivos [Cunha Neto et al., 2020].

Este trabalho tem por objetivo avaliar as limitações de disponibilidade de dados, número de participantes nas rodadas de treinamento federado e probabilidade de falha dos clientes no contexto de redes móveis. Esta análise é motivada pela popularização dos dispositivos móveis, o que faz com que estes constituam a maior parte dos clientes em cenários de aprendizado federado. Logo, os parâmetros de redes móveis, como latência média e probabilidade de falha de conexão, afetam diretamente o desempenho dos modelos treinados [Konečný et al., 2016]. Ademais, devido à natureza limitada das conexões de dispositivos móveis, que normalmente oferecem banda passante inferiores a de conexões cabeadas, aumentar a eficiência no uso dos canais de comunicação é extremamente importante. Tendo isto em mente, este trabalho apresenta um estudo do impacto de parâmetros de rede em cenários federados em métricas comumente utilizadas em redes móveis. Os resultados apresentados são obtidos com o uso de conjuntos de dados e *frameworks* de código aberto e estão disponíveis em um repositório GitHub para permitir sua reprodutibilidade[1]. Inicialmente, a análise cobre uma rede neural convolucional centralizada e a mesma rede neural em um cenário federado com configurações ideais, ou seja, onde parâmetros como falhas de conexão e latência são ignorados. Em seguida, parâmetros de rede mais realistas, que consideram diferentes probabilidades de falhas nos clientes, são analisados e a comparação entre os dois cenários é conduzida. Os resultados obtidos sugerem a necessidade de escolha dos clientes que participam no treinamento como forma de aprimoramento do desempenho do modelo federado.

O restante deste trabalho está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados. A Seção 3 revisa brevemente os conceitos de aprendizado federado e apresenta as particularidades de sua aplicação em redes móveis. A Seção 4 discute a metodologia adotada e a configuração do ambiente experimental. Na Seção 5, os resultados são apresentados e explorados. Por fim, a Seção 6 conclui este artigo e apresenta possíveis direções de pesquisa.

2. Trabalhos Relacionados

Esta seção apresenta trabalhos que avaliam o impacto de parâmetros de rede no desempenho do aprendizado federado. Foram escolhidos trabalhos que apresentam objetivos em comum, como sugerir melhorias em estratégias de agregação de acordo com limitações de rede ou avaliar como certas limitações impostas pelo cenário de rede podem ser contornadas através da variação de parâmetros controláveis. Dentre estes parâmetros estão épocas de treinamento locais ou número de clientes utilizados nas rodadas de treino.

McMahan et al. introduzem o conceito de aprendizado federado, onde o treinamento dos modelos de aprendizado profundo é feito através de uma federação de dispositivos, ao invés da abordagem tradicional, onde os dados são concentrados em um servidor para o treinamento [McMahan et al., 2017]. Os autores descrevem as particularidades do processo de otimização federada e avaliam o desempenho dos algoritmos Média Federada (*FederatedAveraging* – FedAVG) e Descida Estocástica do Gradiente Federada (*FederatedSGD* – FedSGD), além do impacto das rodadas de comunicação no desempenho final

[1] `https://github.com/kaylani2/sbrc2021`.

dos modelos de aprendizado profundo. Apesar da contribuição em propor e avaliar o algoritmo de média federada, os autores apresentam resultados de combinações de hiperparâmetros, especificamente a taxa de aprendizado, obtidas no próprio conjunto de teste, o que provoca a necessidade de reavaliação dos experimentos e de verificação das hipóteses feitas.

Tran et al. abordam o impacto das latências de processamento local e comunicação no tempo total de treino federado [Tran et al., 2019]. Porém, apesar do foco aplicado, os autores consideram um mecanismo em que os dispositivos participantes comunicam atualizações em intervalos programados, o que pode não ser um mecanismo viável para dispositivos não gerenciados diretamente por uma estação base.

Nguyen et al. implementam um sistema para avaliar a presença de *botnets* em redes IoT, onde os *gateways* IoT agem como participantes do aprendizado federado [Nguyen et al., 2019]. Os autores utilizam o desempenho de um modelo centralizado para selecionar a configuração do aprendizado distribuído, porém a análise apresentada avalia apenas a capacidade de detecção dos dispositivos e desconsidera o efeito das rodadas de comunicação do aprendizado federado.

Yang et al. avaliam o impacto de três políticas de agendamento na acurácia e no número de rodadas de comunicação em algoritmos de aprendizado federado em uma rede sem fio [Yang et al., 2020]. O trabalho também avalia como os parâmetros da camada física, por exemplo, a relação sinal-ruído, afetam o número de rodadas de comunicação necessário para garantir a convergência dos modelos federados. Apesar da utilização de políticas de agendamento clássicas, os cenários avaliados assumem plena disponibilidade dos clientes e que apenas o canal de comunicação age como limitador de desempenho, o que pode não representar muitos dos cenários de aprendizado federado em redes sem fio.

Li et al. utilizam o ambiente de desenvolvimento NVIDIA Clara para segmentar imagens de tumores cerebrais [Li et al., 2019]. Os resultados obtidos são comparados com valores simulados em um cenário centralizado. Similarmente, Roth et al. também utilizam o *framework* NVIDIA Clara para a implementação de um sistema para classificação da densidade de seios, com o objetivo de detectar e tratar câncer de mama [Roth et al., 2020]. Os dois trabalhos apresentam seus resultados apenas em comparação direta com o cenário de aprendizado centralizado e utilizam clientes com alta capacidade computacional e configurações homogêneas, além de alta disponibilidade e baixas restrições de conexão. Já Asad et al. avaliam diferentes variações de algoritmos federado, como *Communication-Mitigating Federated Learning* (CMFL) e *Federated Maximum and Mean Discrepancy* (FedMMD) para avaliar seus impactos na eficiência de comunicação, porém apenas um valor para o número de clientes é simulado [Asad et al., 2021].

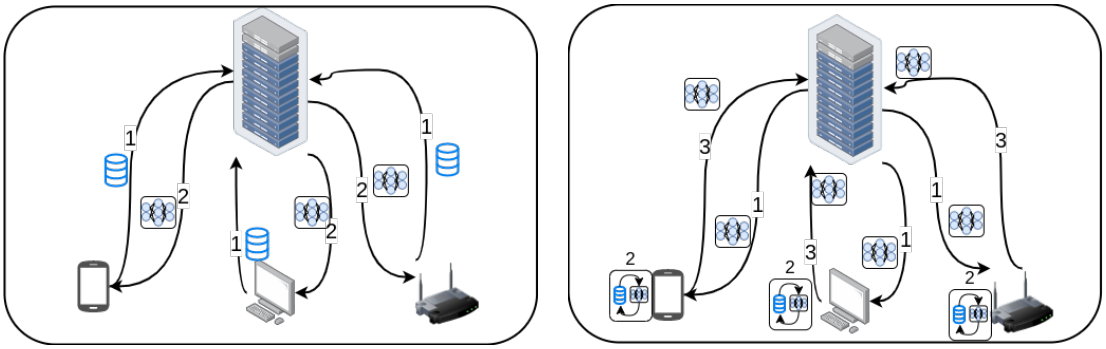
Ressalta-se que existem aplicações de aprendizado federado que se preocupam exclusivamente com a privacidade, logo, os dispositivos possuem alta capacidade computacional e o meio de transmissão não é necessariamente limitado [Li et al., 2019, Roth et al., 2020]. Apesar deste tipo de aplicação estar presente na literatura, este trabalho foca na exploração de redes que operam sob limitações práticas e que são mais representativas do panorama federado em redes móveis. Kairouz et al. apontam a necessidade de simulações reprodutíveis como um problema em aberto no campo de aprendizado fe-

derado [Kairouz et al., 2019]. Existem estudos na literatura que avaliam o desempenho de modelos federados com o uso de máquinas dedicadas para a criação de modelos de aprendizado profundo [Li et al., 2019, Roth et al., 2020], porém essas configurações não representam o cenário de redes móveis, onde os clientes não são capazes de treinar modelos de aprendizado com bilhões de parâmetros.

Diferentemente das outras propostas, este trabalho ataca o estudo de parâmetros de rede dentro do contexto de redes móveis, onde o número de clientes disponíveis é variável, o volume de dados para treino é desconhecido *a priori* e falhas de conexão são mais presentes do que em redes cabeadas.

3. Aprendizado Federado em Redes Móveis

A Figura 1 apresenta a distinção entre o cenário de aprendizado tradicional ou centralizado e o cenário de aprendizado federado. Para realizar o treinamento centralizado, os dispositivos clientes enviam seus dados a um servidor que agrega os dados, treina o modelo de aprendizado e distribui o modelo treinado para que os clientes possam utilizá-lo para fazer inferências. Já no aprendizado federado, o servidor distribui um modelo de aprendizado para os clientes, os clientes treinam os modelos distribuídos em seus dados e transmitem os parâmetros do modelo treinado para o servidor. Esta etapa é repetida diversas vezes e é normalmente chamada de uma rodada de treinamento federado. Entre cada rodada de treinamento, o servidor agrega os modelos de cada cliente e redistribui um novo modelo. Este modelo pode ser treinado novamente ou distribuído para fazer inferências.



(a) Cenário de aprendizado tradicional ou centralizado.

(b) Cenário de aprendizado federado.

Figura 1. Diferentes cenários de treinamento. (a) Na etapa 1 os clientes enviam os dados ao servidor e na etapa 2 o servidor devolve os modelos treinados para os clientes. (b) Na etapa 1 o servidor inicializa e distribui um modelo de aprendizado, na etapa 2 os clientes treinam o modelo com seus dados locais e na etapa 3 os modelos são devolvidos ao servidor para agregação.

Observando a Figura 1(b) é possível destacar alguns parâmetros de redes de computadores que podem afetar o desempenho do algoritmo de aprendizado. O poder computacional dos clientes deve ser suficiente para comportar os modelos de aprendizado em memória e para treinar os modelos de aprendizado dentro da janela de tempo determinada pelo servidor. Além disso, a disponibilidade dos clientes pode resultar em um aumento do número de rodadas necessárias para a convergência do modelo. Finalmente, a latência total entre o servidor e os clientes, apesar de não diretamente impactar o desempenho do modelo de aprendizado, obviamente aumenta o tempo de treinamento global.

Tendo em mente como o aprendizado federado contribui para a privacidade de dados, é importante distinguir entre seus dois principais cenários: *cross-silo* e *cross-device*. No aprendizado federado *cross-silo* os clientes são gerenciados por uma única entidade, ou seja, questões como disponibilidade e poder computacional são conhecidas *a priori*. Os dados gerados por clientes ainda não são concentrados em um servidor e o treinamento é feito de forma distribuída. No entanto, como os clientes são controlados diretamente pelo servidor, é possível orquestrar e definir questões como disponibilidade e capacidade de comunicação. Por outro lado, no cenário *cross-device*, os clientes são usualmente compostos por dispositivos móveis e com características heterogêneas. Neste tipo de cenário o servidor deve ser capaz de lidar com múltiplas desconexões, latências maiores e clientes incapazes de participar no treinamento. Naturalmente, o aprendizado federado em redes móveis se aproxima mais do cenário *cross-device*, onde parâmetros de redes sem fio apresentam forte correlação com o desempenho dos modelos de aprendizado. Neste cenário, os hiperparâmetros de redes neurais, como número de épocas de treinamento e tamanho do *batch* de dados afetam diretamente o desempenho do modelo. Caso os dispositivos móveis possuam baixo poder de processamento, é inviável executar o treinamento por várias épocas; já a disponibilidade de memória limita o volume de dados que o dispositivo pode processar durante o treinamento; por fim, o tempo de conectividade limita o volume de dados a serem transferidos. Logo, este trabalho avalia como alguns desses parâmetros afetam o aprendizado federado, especialmente no contexto de redes móveis, onde a dinamicidade da rede e a heterogeneidade dos dispositivos se apresenta como uma área inexplorada e como um fértil campo de pesquisa.

4. Metodologia e Configuração dos Experimentos

Uma única arquitetura de rede neural foi utilizada em um treinamento centralizado com o objetivo de estabelecer resultados base para a comparação com o cenário federado. O resultado base foi obtido com um mecanismo de parada antecipada para calcular o valor ótimo de classificação das amostras centralizadas [Li et al., 2020]. As métricas de classificação foram usadas para estimar, de forma qualitativa, se o algoritmo de aprendizado federado convergiu. O algoritmo centralizado, após ajuste de hiperparâmetros, atingiu acurácia de $67,82\%$ no conjunto de teste. Este valor foi usado como "teto" para estimar se o algoritmo distribuído convergiu. Todos os experimentos foram conduzidos utilizando o conjunto de dados CIFAR-10 [Krizhevsky, 2012]. Este conjunto para classificação de imagens foi escolhido por ser largamente utilizado em outros trabalhos da literatura para avaliar o desempenho de aplicações em redes. Os códigos foram desenvolvidos sobre o *framework* `Flower` [Beutel et al., 2020] com o *backend* `TensorFlow`[2].

É importante mencionar que o tamanho do modelo de aprendizado escolhido, isto é, o volume de dados transferido para os clientes, deve ser limitado devido às restrições de *hardware* dos dispositivos no cenário móvel. Mesmo que, em aplicações reais, os clientes participem do treinamento apenas durante períodos de baixa atividade, é importante que os consumos de memória, CPU e GPU do modelo não atrapalhem o funcionamento de outras aplicações. Apesar de ser possível atingir melhor desempenho com uma rede neural mais profunda, o custo de comunicação do modelo poderia torná-la inviável, sendo que a acurácia inferior do modelo centralizado utilizado não prejudica a generalização

[2]Acessado em `https://www.tensorflow.org/`.

dos métodos distribuídos analisados na Seção 5. O uso de modelos mais profundos apenas altera o "teto" utilizado para avaliar a convergência do modelo distribuído. Logo, a avaliação desses modelos para outros cenários de rede está fora do escopo deste artigo e é definida como uma possível futura linha de pesquisa. Tendo isso em mente, o modelo final foi limitado para utilizar apenas 550.570 parâmetros, o que corresponde a $4,3$ MB de dados a serem transferidos a cada rodada do aprendizado federado.

Após obtida a acurácia e os hiperparâmetros para a rede neural, simulações foram feitas a fim de obter o número de rodadas de comunicação necessário para atingir convergência sob condições ideais de rede. Neste cenário inicial, os clientes envolvidos no aprendizado possuem probabilidade de desconexão nula, alta largura de banda e capacidade computacional ilimitada. Estes resultados são utilizados para analisar o impacto de certos parâmetros de rede no número de rodadas necessárias para a convergência, a acurácia final do modelo e o tempo real de treinamento federado. Neste ponto, é importante definir o termo rodada como o processo de envio do modelo de aprendizado do servidor para os clientes, o treino local em cada cliente por um determinado número de épocas, o envio dos modelos de aprendizado treinados dos clientes para o servidor e a agregação dos modelos feita pelo servidor. Uma família de curvas é obtida variando os seguintes parâmetros de simulação: número de clientes participantes (N_c), número de épocas de treinamento de cada cliente (E_t) e tamanho dos *batches* locais de cada cliente (B_c). A fim de melhor representar um cenário real de aprendizado federado, cada cliente possui um parâmetro de simulação adicional chamado passos por época, que controla o limite superior de amostras exibidas ao cliente durante o treino. Este novo parâmetro é usado para que os clientes não tenham acesso ao conjunto de dados completo durante o treinamento e seu valor reflita quantos *batches* locais os clientes utilizam por época de treinamento. Nesta etapa o número de passos por época é mantido constante a fim de impedir que os clientes usem o conjunto de dados inteiro para o treinamento. Esta decisão foi tomada visto que é comum que os clientes do aprendizado federado tenham conjuntos de dados locais com distribuições estatísticas variadas.

Finalmente, os parâmetros de rede são limitados a fim de emular um cenário de aplicação móvel, onde os dispositivos clientes podem sofrer desconexões, não conseguir executar todas as épocas de treinamento ou processar grandes quantidades de dados. Cada cliente é instanciado durante a simulação com uma probabilidade de desconexão, probabilidade de executar menos épocas de treinamento e tamanho do conjunto de dados local irregular. A Tabela 1 apresenta os parâmetros de redes avaliados e adianta os valores que serão usados para a obtenção dos resultados apresentados na Seção 5.

5. Resultados

Esta seção apresenta os resultados obtidos a partir das simulações descritas na Seção 4. Um computador Intel Core i$5-10400$ $2{,}90$ GHz com 6 núcleos de processamento e com 32 GB RAM foi utilizado durante o treinamento. Os hiperparâmetros da rede neural definidos pelo treinamento centralizado podem ser vistos no Código 1. A definição da arquitetura foi feita utilizando os princípios de construção dos modelos VGG, com número de canais progressivamente maior [Simonyan e Zisserman, 2015, Brownlee, 2018]. A rede neural, como consequência direta da arquitetura utilizada, possui apenas 550.570 parâmetros e o arquivo em formato *Hierarchical Data Format* possui $4,3$ MB. As DNNs são distribuídas para treinamento nos clientes e, a cada rodada, a acurácia é avaliada em

Tabela 1. Parâmetros que impactam o desempenho do aprendizado federado, símbolos utilizados, significados e valores estudados.

Símbolo	Significado	Valores apresentados na Seção 5
N_c	Número de clientes	[5, 10, 20]
E_l	Épocas de treinamento local	[10, 20]
B_c	Tamanho dos *batches* locais de cada cliente	[64, 256]
P_f	Probabilidade de falha de um cliente	[0%, 25%, 50%, 75%]
L_c	Latência de cada cliente	Valores simulados
R	Número de rodadas de treinamento federado	[1..250]

um conjunto de teste em 2 clientes sorteados dentre os participantes.

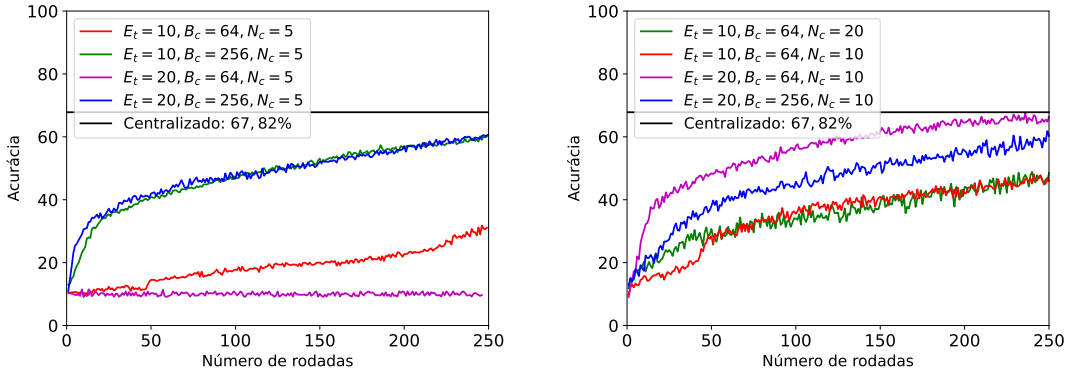
Código 1. Arquitetura da rede neural.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 32)        896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 64)        36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 128)         73856
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 128)         147584
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 128)               262272
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
Total params: 550,570
```

5.1. Condições Ideais de Rede

Neste cenário as condições da rede são ditas ideais, logo, não há atraso de transmissão, não há probabilidade de desconexão e os clientes sempre enviam os pesos corretamente ajustados para o servidor. Observando a Figura 2, pode-se notar que o número de épocas locais de treinamento impactam positivamente a capacidade de convergência do algoritmo. Isto ocorre já que, para um mesmo número de rodadas de treinamento e clientes participantes, mais épocas de treinamento resultam em mais dados utilizados para o treino no modelo global. Ademais, o tamanho dos *batches* locais que assumem um papel regularizador no aprendizado profundo centralizado [LeCun et al., 2015], no aprendizado federado, acelera o processo de convergência. Isto se deve ao fato de que, como cada cliente é limitado a um número fixo de passos por época, um *batch* maior permite que cada cliente utilize mais dados em cada rodada do treinamento federado.

Outro parâmetro importante para o aprendizado federado pode ser observado ao comparar as Figuras 2(a) e 2(b). O número de clientes contribui diretamente para a convergência e para o desempenho do modelo. Entretanto, em cenários móveis, o número de clientes disponíveis para o treinamento é limitado e variável. Tendo isso em mente, técnicas para avaliar se o servidor deve executar uma rodada de treinamento para um dado grupo de clientes devem ser desenvolvidas. A curva azul na Figura 2(b) mostra que o desempenho do modelo pode cair caso haja um número insuficiente de clientes e o servidor deve ser capaz de regular dinamicamente o treinamento. No entanto, como pode ser visto na Figura 2(a), é possível que o modelo obtenha um bom desempenho, porém o treinamento se torna muito mais sensível aos parâmetros do aprendizado, demonstrado pelo desempenho reduzido ao se utilizar 10 épocas de treinamento. Neste resultado o tamanho dos *batches* locais contribui menos para a diferença de desempenho.



(a) Curvas obtidas com 5 clientes. (b) Curvas obtidas com 10 e 20 clientes.

Figura 2. Convergência do modelo de aprendizado federado para condições de rede ideais.

5.2. Condições Reais de Rede

Neste cenário os parâmetros de rede como latência e probabilidade de desconexão são considerados. Os resultados obtidos são comparados com os encontrados para condições ideais de rede, a fim de melhor aproximar o comportamento de dispositivos sem fio móveis.

Probabilidade de desconexão: para casos onde a latência de um cliente ultrapassa o tempo limite de uma rodada, o cliente é ignorado durante aquela rodada de treinamento, ou seja, o servidor infere que o cliente foi desconectado durante o treinamento. Para fins de simulação, caso um cliente sofra uma falha, o servidor ignora a participação do cliente com falha na rodada atual de treinamento. Consequentemente, a convergência do modelo de aprendizado é prejudicada com o aumento na probabilidade de falha nos clientes. Os valores simulados foram escolhidos para representar casos de falha críticos e não críticos. Vale também mencionar que o desempenho do modelo federado é similar para probabilidades de desconexão dentro da faixa entre 5% e 25%.

A Figura 3 apresenta um dos modelos vistos na Figura 2(b), porém com diferentes probabilidades de falha (P_f) aplicadas a todos os clientes. Nota-se que um grande número de desconexões prejudica o desempenho do algoritmo e sua capacidade de convergência. A curva vermelha ($P_f = 25\%$) na Figura 3 mostra que o algoritmo é capaz de convergir em casos de desconexão, porém atinge desempenho inferior. No entanto, ao menos nos cenários estudados, dependendo do número de clientes participantes, o efeito sobre a convergência nas primeiras rodadas pode ser maior ou menor. Isto pode ser visto na curva azul ($P_f = 75\%$) que apresenta desempenho superior ao da curva verde ($P_f = 50\%$), mesmo tendo em vista que mais clientes apresentam falha durante o treinamento. Este resultado pode ser explicado pelo efeito de regularização introduzido pela ausência de alguns clientes durante determinadas rodadas de treinamento, analogamente a como uma camada de *dropout* pode reduzir a dependência do modelo sobre alguns neurônios e aumentar seu desempenho. No entanto, este resultado deve ser mais explorado para melhor sedimentar uma possível conclusão.

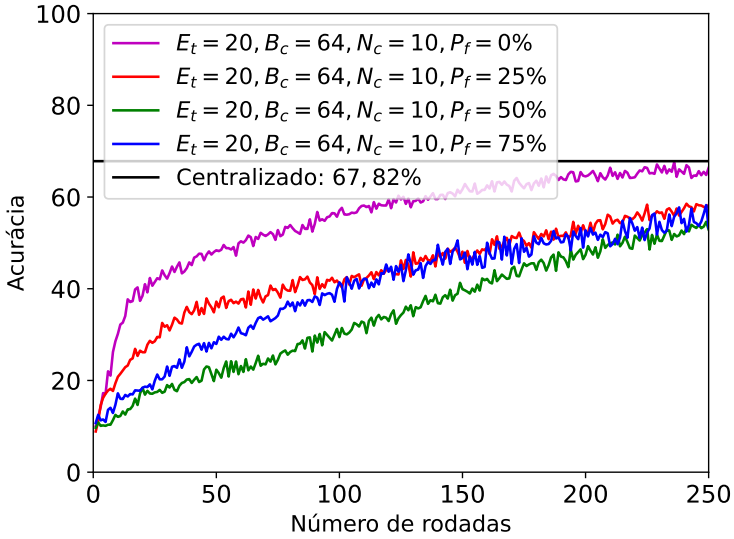


Figura 3. Impacto da probabilidade de desconexão durante o treinamento federado.

Latência: através dos resultados obtidos ao considerar condições de rede ideais, é possível estimar o impacto de conexões limitadas no tempo de convergência do modelo federado. É importante notar que, para cenários em que o tempo de processamento e transmissão de um cliente não ultrapassa o tempo limite de uma rodada de treinamento,

esta latência introduzida aumenta o tempo necessário para a convergência do algoritmo, porém não impacta o número de rodadas diretamente. Este impacto pode ser obtido segundo a Equação 1 a seguir:

$$\text{Latência total} = \sum_{i=1}^{R} \max_{\forall c \in C_I} (\min(L_c, T)), \tag{1}$$

onde C_i representa o conjunto de clientes selecionados para uma rodada de treinamento, L_c representa a latência de um cliente em uma rodada de treinamento e T é um parâmetro de *timeout* configurado pelo servidor. Nesta modelagem a latência de um cliente (L_c) é determinada pela soma entre o tempo de processamento local e os tempos de transmissão entre servidor e cliente. É importante notar que a probabilidade de desconexão tem um impacto direto na latência total, basta 1 cliente apresentar falha para que o tempo da rodada seja igual a T.

A Figura 4 apresenta uma simulação da Equação 1 para diferentes probabilidades de falha. O *timeout* (T) foi escolhido como o dobro do pior caso de latência dentre os dispositivos e naturalmente deve ser ajustado de acordo com a aplicação. A latência de transmissão foi estimada utilizando a ferramenta `iPerf3`[3] para medir o tempo de transmissão até 3 servidores no Brasil, no Canadá e na Rússia. A origem de cada cliente era sorteada entre os 3 servidores. Além disso, o tamanho do modelo transmitido entre clientes e servidor é definido pelo número de parâmetros da rede e é igual a $4,3$ MB. O tempo de processamento dos dispositivos foi estimado através das simulações feitas anteriormente. Nota-se que uma possível estratégia para reduzir o impacto de clientes falhos é encerrar uma rodada de treinamento de forma antecipada, caso uma fração mínima dos clientes comuniquem seus resultados ao servidor. Assim, o servidor pode reiniciar a comunicação com os clientes anteriores ou escolher um novo subconjunto para reduzir a latência. Esta estratégia está representada na Figura 4, onde o tempo total de treinamento é reduzido para um grande intervalo de probabilidades de falha (P_F). É necessário mencionar que há um compromisso entre o tempo total de treino e o custo computacional nos dispositivos, visto que a fração do treinamento feita nos dispositivos mais lentos não é aproveitada.

A Figura 5 resume o número de rodadas para convergência de algumas configurações avaliadas. Os triângulos apontando para a direita representam modelos que apresentavam sinais de melhora no desempenho em 250 rodadas, enquanto $\times$ representa um modelo cujo desempenho piorou com o passar das rodadas de treino. A convergência de um modelo é inferida verificando se a acurácia no conjunto de teste se mantém estável por diversas rodadas de treinamento. Nela, observa-se que altas taxas de desconexão, além de atrasar a convergência do algoritmo, também podem prejudicar o desempenho final do modelo. Aliado ao contexto introduzido pela Figura 2(a), pode-se inferir que, caso o número de desconexões faça com que o mesmo grupo de clientes contribua para o treinamento durante diversas rodadas, o desempenho do algoritmo pode até mesmo piorar com o passar das rodadas. Neste caso, o servidor deve identificar a queda de desempenho e cessar o processo de treinamento até que um número maior de clientes esteja disponível para participar do treinamento. Destaca-se o desempenho superior para o modelo com

[3]Acessado em `https://iperf.fr/`.

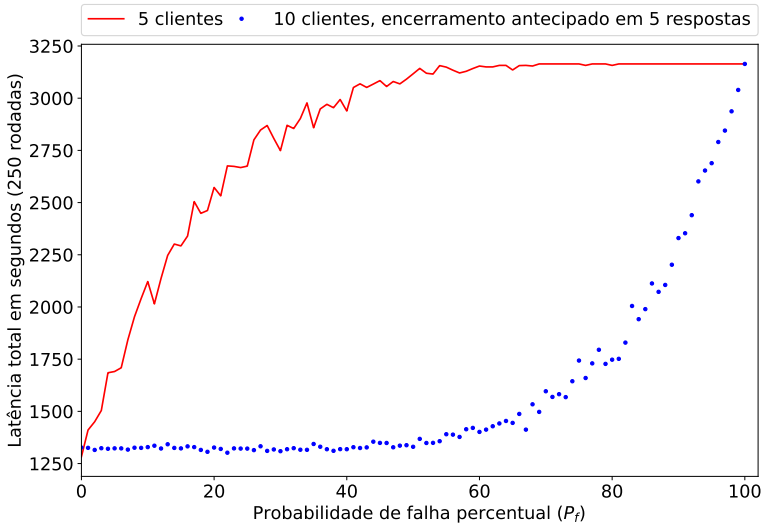


Figura 4. Comparação de latência da Equação 1 com 5 clientes (esquerda) e com 10 clientes participantes, porém com encerramento antecipado de rodada ao atingir 5 respostas dos clientes.

$P_f = 75\%$ em relação ao modelo com $P_f = 50\%$. Mesmo com mais clientes falhos durante o treinamento, o modelo atinge melhor resultado e converge mais rapidamente.

6. Conclusão e Trabalhos Futuros

Este trabalho avaliou o impacto de parâmetros de redes de computadores no desempenho de algoritmos de aprendizado federado. O conjunto de dados foi divido entre clientes de forma a simular a natureza aleatória e independente dos dados em um cenário real, onde os padrões de uso de cada usuário geram dados diferentes entre si. Os resultados mostram o potencial do aprendizado federado em manter a privacidade dos usuários e mesmo assim alcançar taxas de acurácia próximas ao caso centralizado.

A dificuldade em obter conjuntos de dados apropriados para a avaliação de algoritmos federados ainda é um desafio. Além dos rótulos utilizados como alvos para os algoritmos de aprendizado, também é necessário que as amostras individuais contenham uma identificação única de usuário. Este novo requerimento diverge dos cenários de geração de dados usuais, onde um servidor central é utilizado para simular tráfego de rede. A simulação de dispositivos com comportamentos distintos se apresenta como um desafio e oportunidade de pesquisa para a geração de conjuntos de dados apropriados, o que contribuiria para a sedimentação dos resultados obtidos.

Como trabalho futuro, pretende-se construir um sistema para automatizar a coleta de dados nos dispositivos clientes e utilizar dados reais em diferentes configurações de rede para solidificar os resultados obtidos. Como mencionado na Seção 4, pretende-se ainda avaliar o impacto de modelos mais profundos nas configurações avaliadas, tendo em mente cenários de baixa latência e com maior poder computacional na borda, como em redes veiculares e industriais. Ademais, também pretende-se estudar diferentes arquiteturas de redes neurais convolucionais para comparar suas viabilidades. Finalmente, também pretende-se utilizar modelos de mobilidade reais para a análise de desempenho do algoritmo de aprendizado federado.

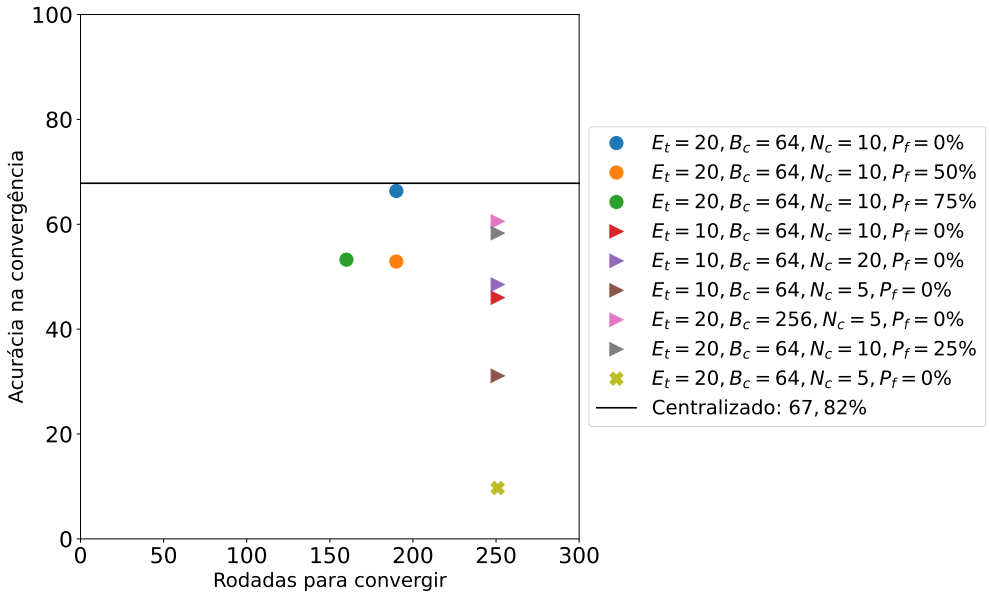


Figura 5. Número de rodadas necessário para que o algoritmo federado atinja convergência.

Agradecimentos

O presente trabalho foi realizado com apoio do CNPq; da FAPERJ, processos E-26/211.144/2019 e E-26/202.689/2018; da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior Brasil (CAPES), Código de Financiamento 001; da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), processo nº 15/24494-8; e da RNP.

Referências

Asad et al. (2021). Evaluating the communication efficiency in federated learning algorithms. Em *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, p. 552–557.

Beutel et al. (2020). Flower: A friendly federated learning research framework. *ArXiv*, abs/2007.14390.

Bochie et al. (2020). Aprendizado profundo em redes desafiadoras: Conceitos e aplicações. Em *Minicursos do XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*.

Brownlee, J. (2018). *Better Deep Learning*. Jason Brownlee, v1.8 edição.

Cunha Neto, H., Menezes, D. e Fernandes, N. (2020). *Privacidade do Usuário em Aprendizado Colaborativo: Federated Learning, da Teoria à Prática*, capítulo 3. Sociedade Brasileira de Computação (SBC).

Dwork, C. e Roth, A. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407.

Hernandez Marcano et al. (2019). On fully homomorphic encryption for privacy-preserving deep learning. Em *2019 IEEE Globecom Workshops (GC Wkshps)*, p. 1–6.

Kairouz et al. (2019). Advances and open problems in federated learning. Em *Foundations and Trends in Machine Learning*.

Konečný et al. (2016). Federated learning: Strategies for improving communication efficiency. Em *NIPS Workshop on Private Multi-Party Machine Learning*.

Krizhevsky, A. (2012). Learning multiple layers of features from tiny images. *University of Toronto*.

LeCun, Y., Bengio, Y. e Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Li, M., Soltanolkotabi, M. e Oymak, S. (2020). Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. Em Chiappa, S. e Calandra, R., editores, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, p. 4313–4324. PMLR.

Li et al. (2019). Privacy-preserving federated brain tumour segmentation. Em Suk, H.-I., Liu, M., Yan, P. e Lian, C., editores, *Machine Learning in Medical Imaging*, p. 133–141, Cham. Springer International Publishing.

Lim et al. (2020). Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22(3):2031–2063.

McMahan et al. (2017). Communication-efficient learning of deep networks from decentralized data. Em Singh, A. e Zhu, J., editores, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, p. 1273–1282, Fort Lauderdale, FL, USA. PMLR.

Narayanan, A. e Shmatikov, V. (2008). Robust de-anonymization of large sparse datasets. Em *2008 IEEE Symposium on Security and Privacy (sp 2008)*, p. 111–125.

Nguyen, P. Q. (2011). Breaking fully-homomorphic-encryption challenges. Em Lin, D., Tsudik, G. e Wang, X., editores, *Cryptology and Network Security*, p. 13–14, Berlin, Heidelberg. Springer Berlin Heidelberg.

Nguyen et al. (2019). DÏot: A federated self-learning anomaly detection system for IoT. Em *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, p. 756–767.

Roth et al. (2020). Federated learning for breast density classification: A real-world implementation. Em Albarqouni, S., Bakas, S., Kamnitsas, K., Cardoso, M. J., Landman, B., Li, W., Milletari, F., Rieke, N., Roth, H., Xu, D. e Xu, Z., editores, *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning*, p. 181–191, Cham. Springer International Publishing.

Shokri, R. e Shmatikov, V. (2015). Privacy-preserving deep learning. Em *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, p. 1310–1321, New York, NY, USA. Association for Computing Machinery.

Simonyan, K. e Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. Em Bengio, Y. e LeCun, Y., editores, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Tran et al. (2019). Federated learning over wireless networks: Optimization model design and analysis. Em *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, p. 1387–1395.

Yang et al. (2020). Scheduling policies for federated learning in wireless networks. *IEEE Transactions on Communications*, 68(1):317–333.