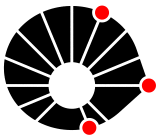

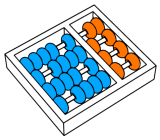

Aprendizado Federado Hierárquico

B. Rosano *L. Bittencourt* *J. Anjos*

Relatório Técnico - IC-PFG-22-03
Projeto Final de Graduação
2022 - Julho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Aprendizado Federado Hierárquico

Bruno Rosano Julio Anjos Luiz F. Bittencourt

Resumo

Cada vez mais dispositivos eletrônicos têm se tornado mais conectados e inteligentes, gerando uma grande quantidade de dados. Com isso, o Aprendizado Federado possibilita explorar esse grande banco de dados distribuído de forma inteligente e mantendo a privacidade dos dados, já que eles não precisam sair do dispositivo que os gerou, ao contrário do que geralmente ocorre nas técnicas de aprendizado de máquina tradicionais. Neste trabalho, é proposta a utilização de um modelo de Aprendizado Federado Hierárquico visando diminuir problemas de comunicação existentes no Aprendizado Federado Tradicional. Primeiramente, descrevemos a implementação de um *framework* para possibilitar criar essa rede hierárquica e posteriormente mostramos os resultados iniciais de testes de desempenho realizados no modelo proposto comparados ao tradicional.

1 Introdução

Devido o crescimento da internet das coisas e a popularização de dispositivos inteligentes, como celulares, equipamentos eletrônicos estão cada vez mais conectados e gerando uma grande quantidade de dados[1], o que possibilita que esses dados sejam utilizados em conjunto com técnicas de aprendizado de máquina para que esses sistemas inteligentes consigam responder questões complexas sobre eles mesmos de forma totalmente independente.

Um problema que decorre da utilização dessa grande quantidade de dados em sistemas de *machine learning* é a privacidade dos usuários [1, 2, 5] é que esses dados podem conter informação sensível e não é desejável que sejam treinados de forma centralizada em um sistema externo como seria proposto para grande partes dos algoritmos de *machine learning* nos dias de hoje. Com isso, técnicas como Aprendizado Federado, onde os dados gerados podem ser treinados de forma local na borda da rede e os apenas os resultados desse treinamento são passados para fora do equipamento mantendo todos os dados gerados locais, têm recebido uma grande atenção, além disso esses dados do treinamento podem ser agrupados com resultados de treinamento de outros equipamentos formando um rede final mais robusta, pois o treinamento está sendo feito em uma quantidade maior de dados mais heterogêneos.

Um exemplo que mostra os benefícios dessa abordagem é a área da saúde [4], onde uma grande quantidade de dados gerada dos pacientes são utilizadas em algoritmos de *machine learning* para ajudar a interpretar sintomas auxiliando na decisão dos próximos passos para o tratamento de um paciente, como os dados gerados são extremamente sensíveis o aprendizado federado pode ser utilizado por fazer com que a privacidade deles seja mantida,

já que nunca saem do ambiente onde foram gerados e ainda podem ser agrupados com dados de outros hospitais o que diminui a chance de que os treinamento seja enviesado pelo composição demográfica dos pacientes em determinado centro médico [4].

No entanto, mesmo com a crescente adesão do 5G no mundo facilitando a quantidade troca de dados entre equipamentos na borda de rede e servidores, os sistemas de FL ainda podem ter problema de comunicação entre os servidores na nuvem e a borda da rede, devido a grande quantidade de dados que são trocados constantemente.

Neste trabalho, é descrita a implementação de uma rede de Aprendizado Federado hierárquica utilizando como base o *framework* Flower[1], onde os equipamentos na borda da rede se comunicam com servidores intermediários que por sua vez se comunicam com a nuvem, ao invés da comunicação direta com a nuvem como é feita na maioria das redes desse tipo buscando aumentar a eficiência e velocidade dessa comunicação. Por fim, mostrou resultados de testes realizados com a rede hierárquica para comparar seu desempenho em relação a rede comum.

2 Referencial Teórico

Antes de entrar na implementação do Aprendizado Federado Hierárquico é necessário entender alguns conceitos que serão utilizados durante este trabalho.

2.1 Aprendizado Federado

Com dispositivos cada vez mais conectados e gerando uma grande quantidade de dados, como celulares, tablets e veículos, a utilização de aprendizado de máquina é de suma importância para explorar esse grande banco de dados de forma inteligente e responder questões complexas para eles mesmos. No entanto, boa parte das soluções de aprendizado de máquina necessitam que os dados sejam agrupados em grandes servidores com capacidade de processamento suficiente para treinar os modelos. Além do grande custo para armazenar e processar esses dados de forma centralizada, também existem preocupações em relação à privacidade desses dados, que pode ser perdido ao entregá-los a um servidor externo.

Para ajudar a solucionar esses problemas foi proposto o Aprendizado Federado [2, 5], que é uma abordagem de *machine learning* distribuída onde diversos equipamentos na borda da rede trabalham de forma colaborativa para treinar um modelo de predição compartilhado em cima de dados mantidos localmente por esses equipamentos tirando a necessidade de enviar esses dados para um servidor externo na nuvem e assim mantendo esses dados, que podem ser sensíveis em alguns contextos, privados.

Nesta abordagem os dispositivos na borda da rede baixam o modelo global e realizam uma etapa de treinamento em cima dele gerando um modelo local, após isso os modelos atualizados por todos os clientes são enviados novamente para o servidor central a fim de que os resultados enviados sejam agrupados atualizando o modelo global do servidor e por

[1]Flower, disponível em `https://github.com/adap/flower`

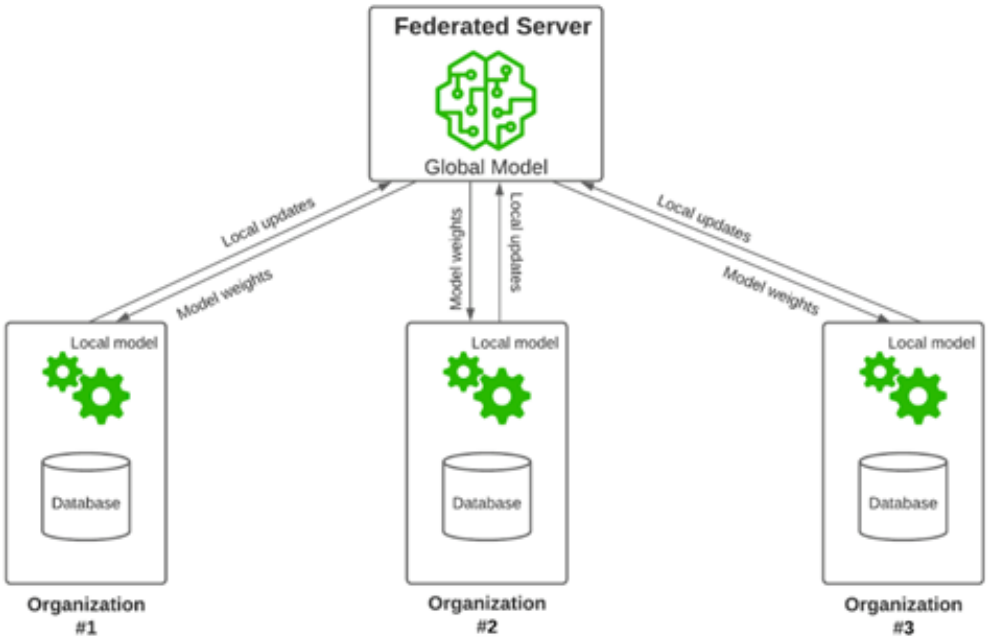

Figura 1: Uma rede de Aprendizado Federado[7]

fim esse modelo central é novamente enviado para a bordas da rede para que os passos possam se repetir para cada rodada de treinamento [2, 3] como pode ser visto na Figura 1.

Durante este trabalho foi utilizado o *Federated Averaging*(FedAvg) [5, 6] como método de agregação, que consiste nos dispositivos realizarem um passo da descida de gradiente com os dados locais para atualizar os modelos e o resultado disso é enviado para o servidor, que realiza uma média ponderada dos resultados para construir o modelo central.

2.2 Latência

Uma das métricas que será utilizada para analisar o desempenho das redes testadas durante o trabalho é a latência. Ela é o tempo que demora entre uma chamada realizada por um cliente e a resposta do servidor retornar para o cliente, em geral esse tempo é medido em milissegundos. Em geral a primeira comunicação pode ter uma latência maior,já que é necessário estabelecer uma comunicação inicial com o servidor e comunicação subsequentes terão menor latência já que a comunicação já foi estabelecida.

Uma rede com alta latência significa que existem diversos atrasos na comunicação, o que afeta o desempenho como um todo, pois mesmo que a operação realizada no servidor seja rápida esta latência torna a atrasa a obtenção do resultado final gerando a percepção de que a operação como todo foi lenta.

2.3 Tipos de Computação

Neste projeto a computação realizada na rede de Aprendizado Federado Hierárquico será dividida em três tipos, descritos nesta seção.

2.3.1 Computação nos dispositivos de borda

Este tipo de computação é realizada diretamente nos dispositivos na borda da rede, como celulares, computadores e tablets. No Aprendizado Federado ele é utilizado para realizar o treinamento dos modelos locais baseado nos dados gerados pelos dispositivos ativos, ou seja, aqueles que têm capacidade de processamento para realizar o treinamento[12].

2.3.2 Computação em nuvem

Outro conceito que teve grande crescimento nos últimos anos é o de computação em nuvem que pode ser definida conjunto de serviços computacionais oferecidos através da rede, esses serviços podem ser poder de processamento, armazenamento, inteligência artificial, entre outros, ou seja, é uma forma de ser acessar recursos computacionais sem possuí-los localmente, apenas acessando eles em servidores de forma remota[8, 9]. Em geral a computação em nuvem é bastante centralizada, possui menor número de equipamentos e possui altas latências em relação a borda da rede e tem grande poder de processamento[10].

No caso de modelos tradicionais de Aprendizado Federado esse tipo de computação é usado para coordenar o treinamento e agregar os modelos enviados pelos clientes, como a comunicação entre a nuvem e a borda é constantes e com uma grande quantidade de dados a latência pode interferir de forma negativa no tempo para se obter resultados de treinamento satisfatórios. Esse é um dos principais problemas que tentamos abordar ao propor um Aprendizado Federado Hierárquico com servidores intermediários na névoa da rede com uma latência menor.

2.3.3 Computação na névoa

Um conceito menos popular que os outros citados, mas que vem ganhando relevância com a popularização de dispositivos *IOT* é a computação na névoa devido ela busca aumentar a eficiência, performance e reduzir a quantidade de dados trocados diretamente com a nuvem. Podemos definir computação na névoa como uma arquitetura de computação geograficamente distribuída de forma descentralizada com dispositivos heterogêneos e onipresentes que distribui recursos e serviços de computação de forma mais próxima da borda da rede[10]. É possível ver a arquitetura dessas camadas da rede na Figura 2.

Este tipo de computação é utilizado no modelo de Aprendizado Federado Hierárquico nas camadas entre o servidor e os clientes, devido a sua maior proximidade a borda da rede o que leva a uma latência reduzida se comparado com a computação em nuvem em geral[10]. No caso de dispositivos passivos na borda da rede, ou seja, aqueles que não tem a capacidade para realizar o treinamento do modelo, os dados gerados por esses dispositivos podem ser transferidos para um servidor-cliente local na névoa que por sua vez realiza o

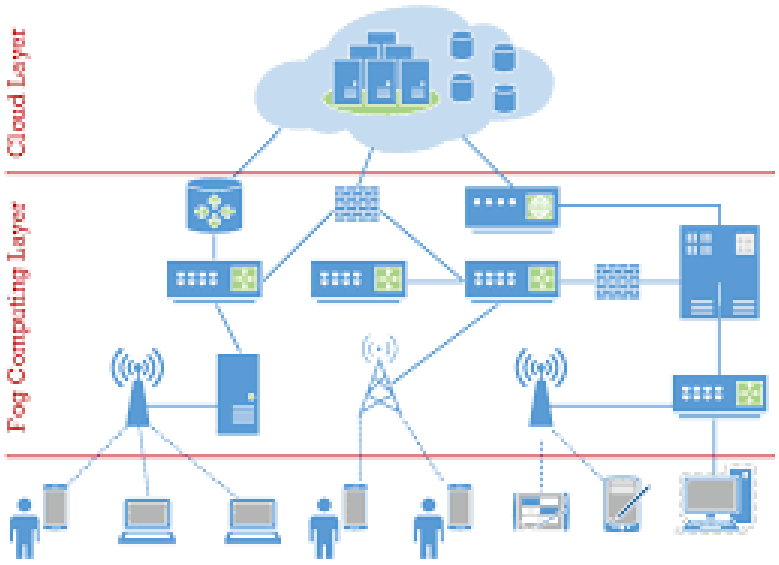

Figura 2: Arquitetura das camadas de borda, névoa e nuvem[11]

treinamento, esse é o caso em alguns sistemas em hospitais que possuem um servidor local central que recebe os dados gerados pelos equipamentos hospitalares[12].

3 Objetivos

Grande parte dos estudos em Aprendizado Federado é feito em cima de redes com os clientes na borda da rede e os servidores na nuvem. Neste trabalho será descrita a implementação de modificações realizadas no *framework* Flower, o que permite testar de forma simples modelos de Aprendizado Federado em dispositivos de borda reais[13]. No entanto, por padrão, assim como a maioria dos *frameworks* nesta área, ele suporta apenas redes com duas camadas.

Com as adaptações feitas ao Flower, foi criada uma rede Aprendizado Federado Hierárquico com 4 camadas, sendo elas: (i) a dos clientes na borda da rede; (ii e iii) duas camadas intermediárias com servidores na névoa da rede; e (iv) uma última camada com o servidor na nuvem. Foram realizados testes preliminares de desempenho com essa rede e com uma rede de 2 camadas tradicionais a fim de entender possíveis vantagens e desvantagens do modelo hierárquico. Os principais aspectos a serem avaliados foram como a rede utilizada afeta o tempo de execução do Aprendizado Federado, além de sua acurácia e função de perda. Entender esses pontos é de grande importância para entender a viabilidade de se adotar uma nova topologia de rede em projetos de grande escala.

4 Metodologia

Para responder às questões levantadas neste trabalho foi necessário implementar um framework que tornasse possível criar a rede hierárquica proposta. Nesta sessão são apresen-

tadas as alterações que foram realizadas no Flower e que permitiram que fossem realizados os testes propostos. Além disso, serão explicadas quais decisões foram tomadas de como montar nosso sistema para os testes.

4.1 Comunicação

O Flower utiliza um ciclo de FL no servidor para coordenar todas as etapas a serem executadas no Aprendizado Federado, esse ciclo pede para a *Strategy*, que é a estratégia de agregação utilizada como FedAvg por exemplo, qual o próximo passo a ser tomado e envia essas configurações aos clientes para executar o passo e após um tempo recebe os resultados das operações dos clientes para serem agregados de acordo com a *Strategy* selecionada[13].

É importante para a coordenação desse sistema complexo definir como será feita a comunicação dos dados entre clientes e servidores. O Flower é agnóstico a stack de comunicação devido seu proxy de cliente(*ClientProxy*), que é uma interface abstrata que encapsula detalhes de como se comunicar com cada cliente, por padrão existe apenas a implementação GrpcClientProxy, onde a comunicação entre o cliente e o servidor é feita utilizando *streams* bi-direcionais de gRPC(*Google Remote Procedure Call*), que funciona bem em redes com baixa largura de banda e permitem que a conexão entre servidor e cliente não precise ser restabelecida a cada nova comunicação [13]. Cada cliente conectado é guardado como um *ClientProxy* pelo servidor dentro de um *ClientManager* que permite o servidor escolher quais clientes conectados serão utilizados para algum passo do Aprendizado Federado e a partir do momento que um cliente se torna inativo seu *ClientProxy* é removido[13].

Para as novas camadas implementadas foi mantida comunicação feita por gRPC a fim de se aproveitar dos benefícios trazidos por ela e utilizar dos tipos de mensagens RPC já existentes, eliminando a necessidade de se alterar componentes presentes do *framework*.

4.2 Camadas Intermediárias

Outro passo importante para permitir uma rede hierárquica foi a implementação de camadas intermediárias adicionais. Para isso, foi criada uma nova estrutura dentro do Flower, chamada de combinador, que possui dentro de si um cliente, que se comunica com a camada superior, e um servidor, que se comunica com a camada inferior. Essa nova estrutura foi criada usando como base o modelo de combinador existente no *framework* FedN [2], que permite apenas redes de Aprendizado Federado com três camadas, sendo a intermediária o combinador.

O cliente do combinador se comunica com o servidor da camada superior para receber as instruções dos próximos passos a serem executados, a mensagem recebida é processada e repassada para o servidor do combinador para que ele possa repassar as instruções para os clientes conectados a ele na camada inferior. Após isso, o servidor recebe a resposta dos clientes em relação às instruções que foram passadas, fazem o processamento delas e envia para o cliente do combinador o que deve ser respondido para o servidor da camada acima que por sua vez processa essa mensagem e envia ela com a resposta das instruções passadas.

[2]FedN, disponível em `https://github.com/scaleoutsystems/fedn`

Um exemplo dessa comunicação em uma rede com três camadas, sendo a intermediária o combinador, é o servidor da última camada envia para o combinador um pedido para que seja realizado o treinamento no modelo global. Essa mensagem é processada pelo combinador e enviada para os clientes na camada inferior que por sua vez realizam esse treinamento e enviam os parâmetros dos modelos atualizados para o combinador. Após isso, esses dados são agrupados pelo combinador para gerar um modelo único a partir dos parâmetros dos clientes e enviar esses parâmetros para o servidor da camada superior, que recebe os dados de todos os combinadores conectados a ele e faz a agregação de todos os parâmetros para gerar um modelo global atualizado.

Algo importante para permitir mais de três camadas é que a comunicação entre dois combinadores seja possível. Isso é proporcionado pelos clientes do combinador ao implementarem a interface do *ClientProxy* e os servidores se comunicarem com essas interfaces do *ClientProxy* através de um *ClientManager*. Dessa forma todos os clientes e servidores tradicionais do Flower podem se comunicar com os combinadores sem necessitar qualquer alteração a lógica de comunicação deles e o mesmo vale para a comunicação entre combinadores. Além disso, o combinador utiliza o mesmo conceito de *Strategy* já existente, no entanto apenas a agregação é utilizada pois a coordenação continua sendo feita pelo servidor. A arquitetura do Flower após a implementação do combinador, mostrando também a comunicação entre camadas, pode ser vista na Figura 3. Por fim, a implementação do combinador no Flower pode ser encontrada em `https://github.com/brunorosano/flower`.

4.3 Redes Utilizadas

Com todos os componentes necessários implementados foi necessário decidir quais redes seriam utilizadas para os testes do Aprendizado de Máquina Hierárquico e Tradicional, levando em consideração os equipamentos disponíveis. Para o primeiro experimento foi criada uma rede com 4 camadas, onde a camada inferior possui 4 clientes, a segunda camada possui 2 combinadores, a terceira camada possui 1 combinador e a última camada possui 1 cliente. As conexões entre as estruturas de cada camada podem ser vistas na Figura 4. Já para o segundo, foi criada uma rede de 2 camadas com 4 clientes na primeira e 1 servidor na segunda de forma que a única diferença entre as duas redes fossem as novas camadas introduzidas. conforme ilustrado na Figura 5.

4.4 Ambiente de testes

Esta seção apresenta a configuração do Aprendizado Federado para os testes realizados. Decidimos utilizar o Pytorch para criar uma rede neural convolucional, descrita pela Figura 6, para os treinos e para obter o *dataset* de treino CIFAR10, que consiste em 60000 imagens 32x32 divididas igualmente entre 10 classes, para cada teste realizamos os treinamentos por 5 épocas e com uma taxa de aprendizagem de 0.003. O treinamento nos clientes foi feito utilizando GPU através do suporte para CUDA existente no Pytorch. Ademais, escolhemos como estratégia de agregação para todos os servidores e combinadores nas duas redes o FedAvg.

Os testes da rede hierárquica foram realizados utilizando 2 notebooks e duas máquinas

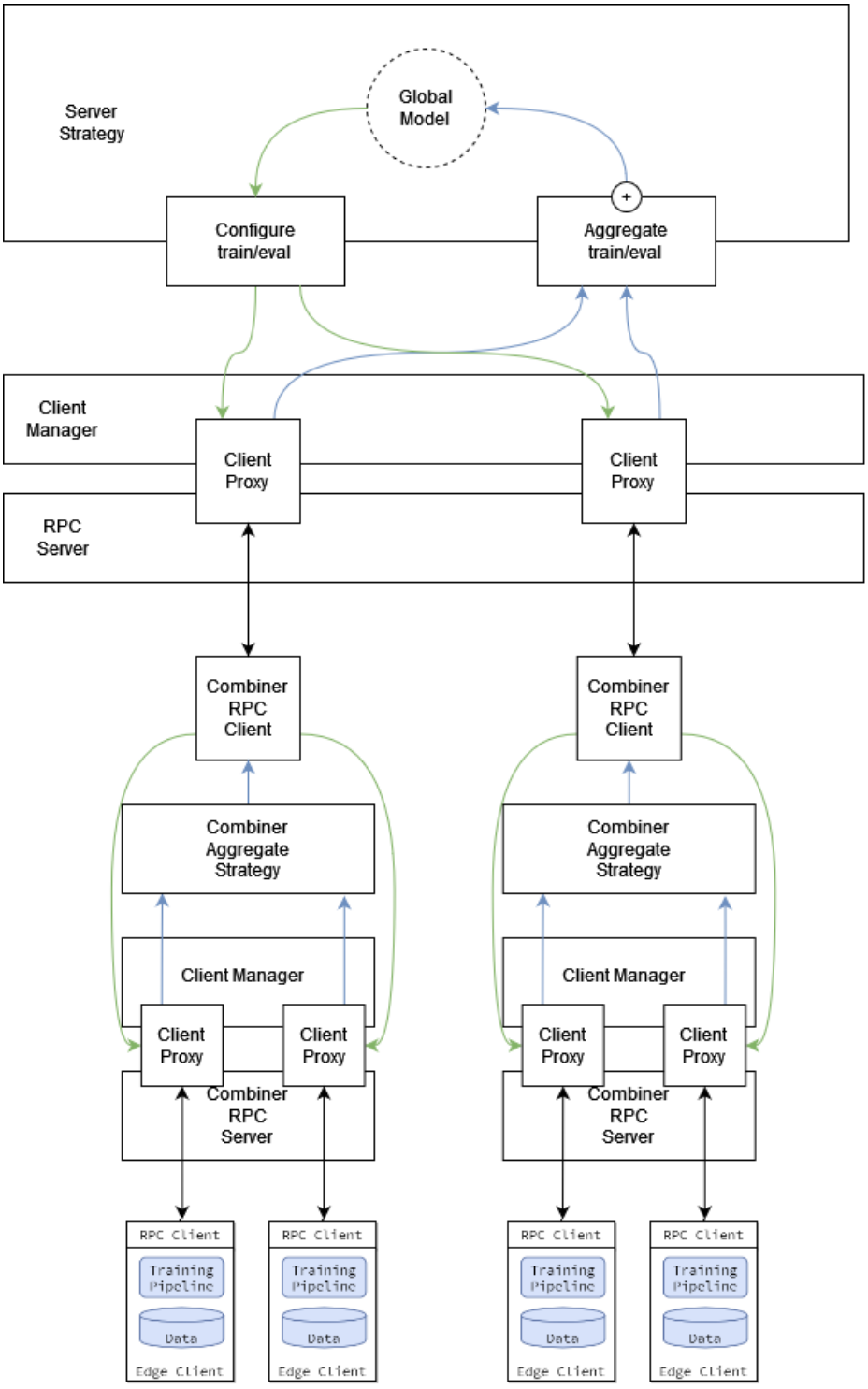

Figura 3: Arquitetura do Flower com o combinador.

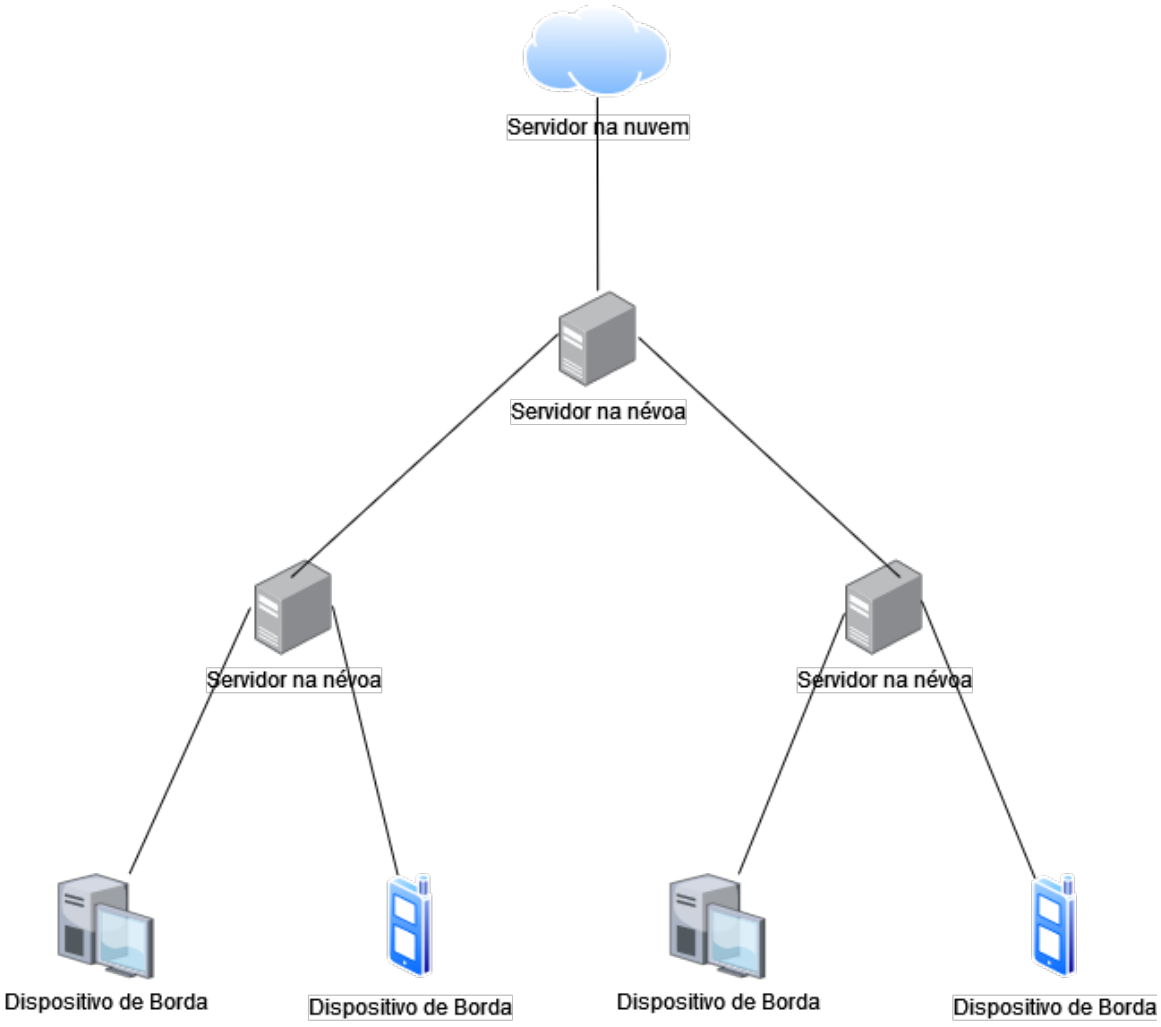

Figura 4: Rede com 4 camadas utilizada para Aprendizado Federado Hierárquico

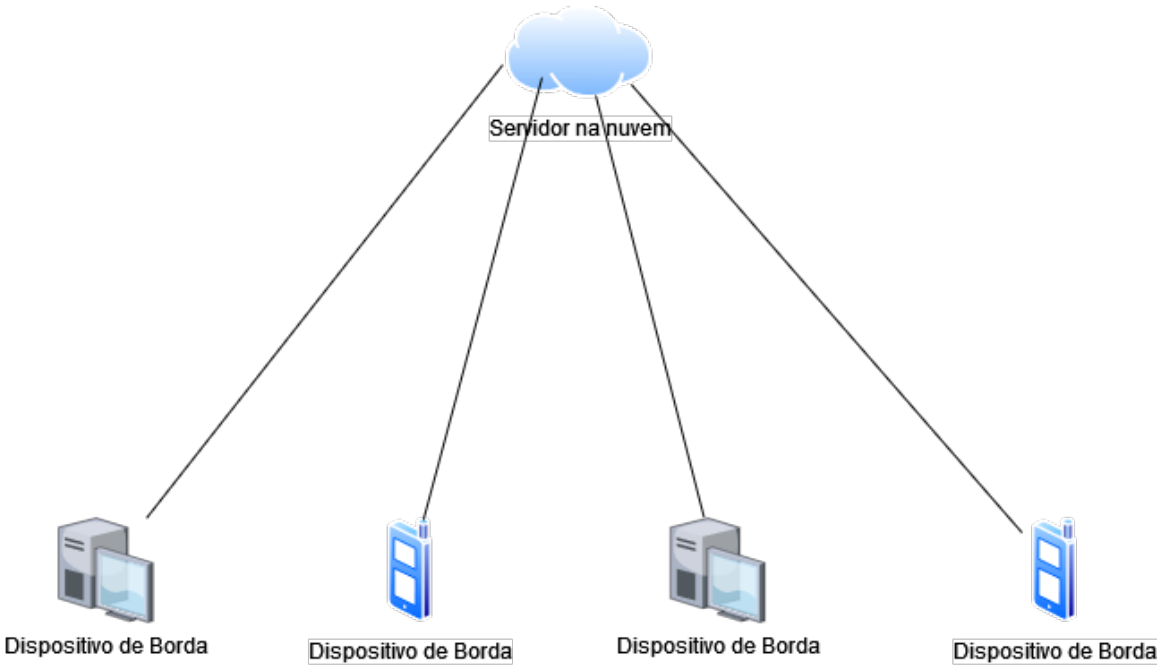

Figura 5: Rede com 2 camadas utilizada para Aprendizado Federado Tradicional

virtuais em uma rede LAN, para minimizar as latências entre os computadores. Para a rede tradicional foram utilizados apenas os 2 notebooks. Em ambos os casos uma máquina era responsável por exatamente uma das camadas utilizadas.

Como todas as máquinas estavam na mesma rede, foi necessário introduzir uma latência sintética entre elas para simular um ambiente real com cada máquina em locais distintos. A fim de se obter esse efeito, utilizamos o *Linux Traffic Control*, que permite configurar o agendador de pacotes do kernel Linux, para criarmos filtros de rede com determinados atrasos na comunicação entre os IPs das máquinas conectadas entre si. O tempo de atraso foi escolhido de forma aleatória dentro de um intervalo selecionado em cada teste realizado para representar diversas topologias de rede possíveis com diferentes níveis de carga. Os intervalos de latência escolhidos para cada camada podem ser vistos na Tabela 1.

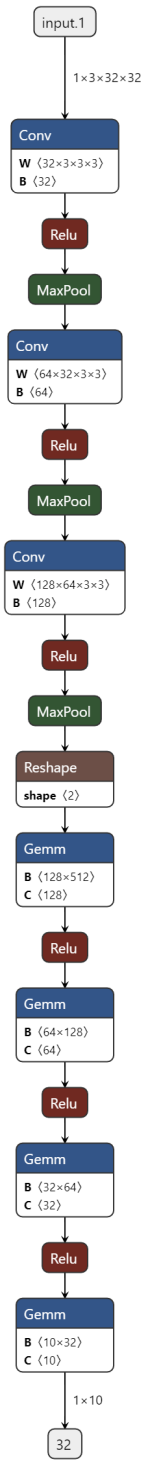

Figura 6: Rede neural utilizada durante os experimento

Intervalo de atrasos entre camadas		
Camadas	Rede Hierárquica	Rede Tradicional
1-2	[2-15]	[42-165]
2-3	[20-50]	-
3-4	[20-100]	-

Tabela 1: Intervalo de atraso entre camadas em milissegundos para a Rede Hierárquica e Tradicional

5 Resultados

Com todas as configurações feitas, realizamos 30 execuções em cada uma das duas redes para avaliarmos diferentes atrasos provenientes de variações nas topologias das redes, pois eram gerados novos valores de latência para a rede a cada nova rodada de testes. Após isso, agrupamos os valores dos testes de cada rede para que fosse realizada uma análise nos resultados obtidos do tempo de execução, da acurácia e da função de perda para a rede hierárquica e tradicional.

5.1 Tempo de execução

Analisando os dados agrupados do tempo de execução presentes na Tabela 2 é possível perceber que a nossa rede hierárquica obteve resultados melhores em todos os pontos analisados, sendo que a média encontrada foi cerca de 2,86% mais baixa, enquanto a mediana foi 1,8% menor. Algo que chamou a atenção foi o alto desvio padrão da Rede Tradicional, onde foi possível perceber que existem dois valores de tempo de execução de cerca de 152 segundos enquanto o terceiro maior valor é de 141 segundos. Esses valores podem ter sido causados também por limitações no hardware, pois a maior parte do tempo de execução é de processamento e não é esperado que apenas a latência resulte em valores tão altos. Removendo esses dois valores *outliers*, o desvio padrão da Rede Tradicional diminui para 4.701, sua média para 132.636 e a mediana para 131.982, no caso das proporções em relação a Rede Hierárquica para a média e mediana ficaram em 1,8% e 1,3%, respectivamente.

Portanto, essa análise, apesar de não conclusiva, mostra que o modelo proposto é promissor em relação ao tempo de execução, visto que houve uma disparidade perceptível entre as duas redes. Já era esperado que a diferença proporcional encontrada fosse baixa, já que, como citado anteriormente, o tempo gasto com comunicação é baixo se comparado com o tempo de processamento para o treinamento da rede.

Tempo de execução em 30 testes		
Tempo de execução	Rede Hierárquica	Rede Tradicional
Média	130.233	133.969
Desvio Padrão	4.067	6.807
25%	127.767	130.182
50%	130.181	132.47
75%	133.491	136.880
Mínimo	121.100	122.389
Máximo	136.277	152.912

Tabela 2: Tempo de execução em segundos para 30 testes
para a Rede Hierárquica e Tradicional

5.2 Acurácia e Função de Perda

Em relação a acurácia das redes é possível perceber pelos dados da Tabela 3 que os resultados foram extremamente próximos sendo que a média e a mediana da Rede Hierárquica foram apenas 0.55% maiores em relação a da Rede Tradicional. Ao analisar a função de perda com os dados descritos na Tabela 4, é possível observar um padrão similar, onde a diferença entre os dois é pequena, sendo que a média da Rede Hierárquica foi 0,5% maior enquanto a mediana foi 0,6% maior em relação a Rede Tradicional.

Nesses casos as latências não possuem influência em relação ao resultado final, no entanto foi possível perceber que mesmo com a complexidade adicionada ao treino pela agregação feita com o FedAvg nas duas camadas intermediárias da Rede Hierárquica os resultados de treinamento finais não foram afetados de maneira significativa. Com isso, percebemos que no caso da nossa configuração de redes os valores de acurácia e da função de perda não se depreciaram devido a Rede Hierárquica

Acurácia em 30 testes		
Acurácia	Rede Hierárquica	Rede Tradicional
Média	0.542	0.539
Desvio Padrão	0.018	0.018
25%	0.526	0.534
50%	0.544	0.541
75%	0.5531	0.549

Tabela 3: Acurácia em 30 testes
para a Rede Hierárquica e Tradicional

Função de perda em 30 testes		
Função de Perda	Rede Hierárquica	Rede Tradicional
Média	1.005	1.000
Desvio Padrão	0.137	0.0137
25%	0.998	0.990
50%	1.004	0.998
75%	1.047	1.005

Tabela 4: Função de Perda em 30 testes
para a Rede Hierárquica e Tradicional

6 Trabalhos Futuros

Apesar de trazer resultados promissores, a nossa abordagem possui diversas limitações que podem ser vistas como oportunidades para trabalhos futuros. Nossos testes foram realizados apenas em um ambiente local, com um número reduzido de dados homogêneos e com poucos elementos na rede. Devido a isso, algumas limitações existentes em rede reais não

estavam presentes, como perdas de pacotes e baixas larguras de banda. Além disso, todas as camadas das redes possuem elementos homogêneos, já que todos estavam sendo simulados no mesmo computador sem levar em conta diferenças consideráveis que podem existir entre dois computadores com poder de processamento desiguais, por exemplo.

Algo que pode ser explorado são testes utilizando o modelo de Aprendizado Federado Hierárquico com máquinas em diferentes localidades e com poder de processamento distintos se conectando entre si através de uma rede real, a fim de poder analisar como nosso modelo se desempenha nesse ambiente. Ademais, é possível realizar experimentos com equipamentos passivos na borda da rede, necessitando que os dados sejam transferidos para a névoa ao realizar o treinamento e utilizar dados heterogêneos nos dispositivos da borda de rede, visto que é esperado que os dados gerados por essas máquinas não sejam idênticos.

Além de alterar o ambiente de execução, trabalhos futuros podem utilizar redes neurais complexas, com um elevado número de parâmetros para analisar os resultados onde a largura de banda influência na performance da rede hierárquica. Outra mudança poderia ser a função de agregação utilizada, pois usamos apenas o FedAvg e outras funções podem afetar os resultados finais da rede devido a complexidade introduzida pelas camadas adicionais. Por fim, devido ao Flower ser agnóstico a *stack* de comunicação poderiam ser feitos testes com outras além do gRPC, já que alguns equipamentos mais simples na borda de rede podem necessitar que a comunicação seja realizada de outra forma.

7 Conclusão

Durante este projeto apresentamos um modelo de Aprendizado Federado Hierárquico buscando melhorar alguns aspectos no modelo tradicional, como a eficiência e velocidade da comunicação entre os nós da rede a partir da introdução de camadas intermediárias na névoa que possuem um latência reduzida em relação à borda da rede. A partir desse modelo, adaptamos o *framework* Flower de forma a podermos implementar uma rede hierárquica, devido a maioria dos frameworks de Aprendizado Federado existentes aceitarem apenas redes com duas camadas tradicionais.

Com isso, criamos um ambiente de teste local com uma rede hierárquica e tradicional utilizando o *Linux Traffic Control* para simular diversas topologias de rede que podem existir no Aprendizado Federado e testar como nosso modelo se desempenha em relação ao tradicional. Os resultados encontrados, apesar de não conclusivos, são promissores por mostrarem que não houve queda de desempenho considerável no treinamento do Aprendizado Federado, além de apontarem uma diferença perceptível no tempo de execução no treinamento em relação ao Aprendizado Federado Tradicional.

Em razão dos experimentos terem sido realizados apenas em uma rede local, com uma quantidade pequena de dados homogêneos e de elementos na rede, existem diversas oportunidades para serem exploradas por trabalhos futuros. Eles podem explorar experimentos em redes reais, com diferentes stacks de comunicação e algoritmos de agregação, assim como utilizar modelos de aprendizado de máquina mais complexos treinados com uma quantidade maior de dados mais heterogêneos. Por fim, podemos dizer que este trabalho atingiu seu objetivo de criar um framework que possibilita testes em redes hierárquicas e de analisar

de maneira inicial um novo modelo de Aprendizado Federado.

Referências

[1] Abad, Mehdi Salehi Heydar, et al *Hierarchical federated learning across heterogeneous cellular networks.* ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2020.

[2] Federated Learning: Collaborative Machine Learning without Centralized Training Data, disponível em `https://ai.googleblog.com/2017/04/federated-learning-collaborative.html`

[3] Bonawitz, Keith, et al. *Towards federated learning at scale: System design.* Proceedings of Machine Learning and Systems 1 (2019): 374-388.

[4] Medical AI Needs Federated Learning, So Will Every Industry, disponível em `https://blogs.nvidia.com/blog/2021/09/15/federated-learning-nature-medicine/`

[5] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise 4era y Arcas (2017). *Communication-efficient learning of deep networks from decentralized data.* Artificial Intelligence and Statistics, 1273–1282

[6] Collins, Liam, et al. *FedAvg with Fine Tuning: Local Updates Lead to Representation Learning.* arXiv preprint arXiv:2205.13692 (2022).

[7] Federated Learning: Predictive Model Without Data Sharing, disponível em `https://sparkd.ai/federated-learning`

[8] Qian, Ling, et al. *Cloud computing: An overview."IEEE international conference on cloud computing.* Springer, Berlin, Heidelberg, 2009.

[9] What is cloud computing? An overview of the cloud, disponível em `https://www.atlassian.com/microservices/cloud-computing`

[10] Bonomi, Flavio, et al. "Fog computing and its role in the internet of things."Proceedings of the first edition of the MCC workshop on Mobile cloud computing. 2012.

[11] Gedeon, Julien, et al. "Fog computing: Current research and future challenges."KuVS-Fachgespräch Fog Comput 1 (2018): 1-4.

[12] What Is Federated Learning?, disponível em `https://blogs.nvidia.com/blog/2019/10/13/what-is-federated-learning/`

[13] Beutel, Daniel J., et al. "Flower: A friendly federated learning research framework."arXiv preprint arXiv:2007.14390 (2020).