Mecanismo para Mitigar Ataques de Envenenamento de Modelo no Aprendizado Federado

Marcos G. O. Morais[1,2], Joahannes B. D. da Costa[1], Luis F. G. Gonzalez[1], Allan M. de Souza[1], Leandro A. Villas[1]

[1]Universidade Estadual de Campinas (UNICAMP), Brasil
[2]Universidade Federal de Lavras (UFLA), Brasil

{moraism, jbdc, gonzalez, allanms, lvillas}@unicamp.br

Resumo. *O Federated Learning (FL) é uma técnica distribuída para treinamento de modelos de aprendizado de máquina, em que os dados são processados localmente e apenas parâmetros locais são compartilhados com um servidor de agregação. O fato de que os dados dos clientes são mantidos localmente torna a tarefa de validar os seu parâmetros uma tarefa extremamente difícil, abrindo portas para possíveis ataques de clientes mal intencionados. Esses atacantes influenciam deliberadamente o treinamento, invalidando o modelo resultante, fazendo injeção de dados e até mesmo manipulando os parâmetros dos modelos. Sendo assim, este trabalho apresenta o RAGNAR que se utiliza de métricas calculadas durante o treinamento, para mitigar em tempo real ataques no ambiente de FL. Além disso, o RAGNAR emprega uma estratégia de agregação dos parâmetros de clientes que se adapta dinamicamente, com o objetivo de lidar com a situação atual. Avaliações experimentais demonstram que o RAGNAR reduz significativamente a perda de acurácia (31%) dos clientes participantes do treinamento e mantém bons níveis de acurácia (98%).*

Abstract. *Federated Learning (FL) is a distributed technique for training machine learning models, where data is processed locally and only local parameters are shared with an aggregation server. The fact that client data is kept locally makes validating their parameters an extremely difficult task, opening doors to potential attacks from malicious clients. These attackers deliberately influence training, invalidating the resulting model, injecting data, and even manipulating model parameters. Therefore, this work presents the RAGNAR, which utilizes metrics calculated during training to mitigate real-time attacks in the FL environment. Additionally, RAGNAR employs a parameter aggregation strategy from clients that dynamically adapts to handle the current situation. Experimental evaluations demonstrate that RAGNAR significantly reduces client training accuracy loss (31%) while maintaining high levels of accuracy (98%).*

1. Introdução

O Aprendizado Federado, do inglês *Federated Learning (FL)*, é uma técnica de aprendizado distribuído que agrega robustez e privacidade ao processo de aprendizado de máquina tradicional [Abdulrahman et al. 2021]. A principal vantagem do FL é a salvaguarda dos dados pessoais de cada dispositivo que participa das rodadas de treinamento, visto que nessa técnica dados brutos não são compartilhados diretamente com o servidor central, reduzindo o risco de exposição ou vazamento de dados sensíveis. Pode-se destacar também que o custo computacional é consideravelmente reduzido, pois a etapa

de treinamento é conduzida de maneira distribuída, onde o dispositivo cliente treina localmente e apenas os parâmetros do modelo são compartilhados e agregados de forma centralizada [de Souza et al. 2024].

No FL, o servidor define e gerencia como cada cliente irá participar do processo de treinamento. O processo se inicia com o servidor central enviando o modelo global (serão valores aleatórios ou pode ser inicializado com um conjunto previamente treinado) a cada um dos dispositivos selecionados, utilizando-se de uma ampla gama de técnicas de seleção para otimizar essa seleção para o treinamento. Após isso, o dispositivo cliente irá treinar o modelo com seus dados locais fazendo os ajustes necessários para que venha a convergir. Após o treinamento local, os parâmetros são compartilhados com o servidor, que por sua vez faz a agregação e atualiza o modelo global. O modelo global atualizado será enviado aos clientes novamente, repetindo o processo quantas vezes for necessário [McMahan et al. 2017].

As aplicações práticas do FL são vastas, e destacam-se especialmente em contextos médicos e industriais, onde a sensibilidade e privacidade dos dados são de extrema importância [Jian et al. 2023]. No setor de saúde, por exemplo, podem ser aplicados modelos de aprendizado de máquina, do inglês *Machine Learning (ML)*, para analisar exames médicos e fazer diagnósticos precisos. Isso pode ser feito sem a necessidade de que os dados médicos sejam enviados para dispositivos fora do hospital, garantindo a confidencialidade [Korkmaz et al. 2022]. Na indústria, o FL pode ser aplicado em cenários de análises de manutenção preditiva, otimização e automação de processos, desenvolvimento de novos produtos entre outras funções. O FL tem papel fundamental nesses cenários, onde dados sensíveis podem ser utilizados para o treinamento de modelos de ML dentro da própria estrutura da organização, reduzindo o risco de vazamento de dados.

No entanto, o FL ainda enfrenta diversos desafios em termos de segurança. Um dos principais desafios está relacionado ao ataque de envenenamento de modelo, também conhecido como falsificação de dados, que está direcionado à degradação do modelo global a partir de ataques Bizantinos [Fang et al. 2020, Mehmuda et al. 2023, Parmar et al. 2024]. Nesse cenário, os atacantes atacam deliberadamente o processo de treinamento com a intenção de invalidar os esforços dos demais dispositivos. A degradação pode ser alcançada através da introdução de ruídos, alteração dos parâmetros ou introduzindo elementos que afetam adversamente o modelo global [Liu et al. 2023]. Nesses cenários, participantes maliciosos introduzem intencionalmente informações falsas ou manipuladas durante o treinamento local comprometendo a integridade e a confiabilidade do modelo resultante. Essa ameaça pode ser particularmente prejudicial em situações em que a seleção dos participantes no treinamento federado não é criteriosa, permitindo a inclusão de dispositivos maliciosos [Jagielski et al. 2018].

Nesse sentido, diversas abordagens se propõem a resolver/mitigar os problemas de segurança do FL. Porém, muitas delas acarretam em custos computacionais significativos e apresentam resistência limitada a uma porcentagem de clientes maliciosos. Dentre tais soluções da literatura, pode-se destacar soluções que utilizam repesagem dinâmica dos parâmetros dos modelos, calibrando-os para que fiquem dentro dos limites das demais respostas recebidas. Outro exemplo é a abordagem KRUM [Blanchard et al. 2017a], que considera uma classificação prévia de clientes confiáveis e tais clientes atualizam o modelo global. Porém, essas estratégias limitam o modelo global a ter uma baixa amostragem do conjunto global de dados disponíveis, atrasando sua convergência.

Diante deste cenário, este trabalho apresenta um mecanismo pa**RA** miti**G**ar ataques de enve**N**enamento de modelos em **A**prendizado fede**R**ado, chamado **RAGNAR**. O RAGNAR identifica e mitiga os ataques de envenenamento de modelos durante o processo de treinamento federado através da eliminação do modelo do cliente atacante da fase de agregação. O RAGNAR se aproveita do fluxo já existente em FL utilizando a fase de agregação dos parâmetros, após o treinamento, avaliando a dispersão e evolução dos modelos rodada a rodada. Os parâmetros benignos serão agregados, enquanto os modelos enviados por clientes maliciosos serão descartados. Experimentos realizados demonstram que o RAGNAR é capaz de identificar clientes maliciosos, mantendo o modelo global integro e livre de ataques em todas as rodadas de treinamento.

O restante deste trabalho está organizado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados, destacando os principais ataques relatados pela literatura. A Seção 3 descreve a solução proposta demonstrando o seu funcionamento, enquanto a Seção 4 apresenta os resultados obtidos. E por fim, a Seção 5 apresenta as conclusões e direções para trabalhos futuros.

2. Trabalhos Relacionados

Diversos trabalhos discutem sobre clientes maliciosos e ataques no contexto de FL. Por exemplo, [Sattler et al. 2020] mostram que a presença de um único cliente malintencionado pode comprometer o treinamento, anulando os esforços dos demais participantes da federação e impedindo a convergência do modelo. [Blanchard et al. 2017b] faz uma análise aprofundada sobre o desempenho do FL regular, demonstrando que este realmente não converge na presença de clientes maliciosos. Isso demonstra a incapacidade do modelo tradicional de agregação em identificar e tratar ataques diretos, levantando preocupações relevantes a respeito do processo.

[Varma et al. 2021] apresentam o LEGATO, solução que realiza a filtragem dos parâmetros camada a camada, normalizando os pesos retornados pelos clientes com o objetivo de normalizar a variância do gradiente e assim fortalecer a robustez do modelo em FL. Porém, para calcular a média ponderada dos gradientes, a solução proposta utiliza parâmetros de rodadas anteriores, atrasando a convergência, visto que está reinserindo no processo dados de rodadas anteriores. Desta forma, o LEGATO não está lidando diretamente com os clientes maliciosos, e sim normalizando todos os gradientes retornados, sendo um processo mais longo e custoso. Outro ponto a ser levado em consideração é que o algorítimo faz o balanceamento em parâmetros enviados por clientes legítimos alterando seus valores reais.

[Briggs et al. 2020] utiliza técnica de clusterização para tratar dados não-Independente e Identicamente Distribuídos (IID), alcançando bons resultados. Porém, esse trabalho demonstra uma técnica com alto custo computacional. Os clientes serão divididos em grupos, de acordo com a semelhança dos parâmetros retornados, e treinados em paralelo cada um com seu subconjunto de dados específicos. Isso acarreta em uma divisão dos dados, diminuindo a quantidade de amostras que cada cliente terá disponível para o seu processo de treinamento.

Alguns trabalhos fazem estudos aprofundados sobre algoritmos de agregação robustos, como o *trimmed mean* [Yin et al. 2021], *Krum* [Blanchard et al. 2017a] e *Bulyan* [Mhamdi et al. 2018], demonstrando suas fraquezas diante dos ataques testados durante os experimentos. De forma resumida, tais abordagens funcionam como se segue: *(i) Trimmed Mean* realiza a classificação das grandezas dos parâmetros retirando o maior e menor

valor do treinamento. Após isso, calcula a média novamente usando os parâmetros restantes e assim gera seu modelo global; *(ii) Krum* leva em consideração os parâmetros de um único cliente para fazer a agregação do modelo global. Esse método reduz drasticamente a quantidade de amostras utilizadas pelo cliente; e *(iii) Bulyan* por sua vez, executa repetidas vezes um outro algoritmo robusto (Krum, por exemplo) buscando alcançar uma robustez ótima. Este método tem um custo computacional alto, o que inviabiliza seu uso em operações reais.

Com base na análise do estado da arte, é possível observar que os trabalhos geralmente não são eficazes em identificar possíveis atacantes bizantinos, e até mesmo podem atrasar a convergência do modelo. Estes trabalhos propõem métodos de repesagem e normalização de gradientes, onde todos os demais clientes participantes serão afetados, sem a identificação do atacante. Além disso, é essencial considerar o custo computacional elevado, o que impossibilita escalar tais propostas.

3. RAGNAR

Diante do cenário descrito, nota-se a necessidade do constante desenvolvimento de técnicas e abordagens que assegurem níveis confiáveis de segurança no FL. Este artigo visa contribuir apresentando uma solução que deriva da técnica *FedAvg* [Abdulrahman et al. 2021], aproveitando o fluxo já existente para avaliar os parâmetros retornados pelos clientes durante o treinamento. O objetivo principal da solução proposta é desenvolver uma estratégia que assegure a imunidade do modelo global em face da influência de modelos corrompidos sem que haja falsos acertos e atrasos na convergência do modelo global. Para isso, calcula-se métricas que serão imprescindíveis para a identificação direta de potenciais clientes maliciosos, determinando quais são úteis para a evolução do modelo global.

O RAGNAR opera como um *plugin*, derivando de uma técnica amplamente utilizada para treinamento federado e se tornando versátil na possibilidade de ser incorporado a outras técnicas de seleção de clientes, como as propostas por [Cho et al. 2020] e [de Souza et al. 2023]. Além disso, pode ser utilizado em mecanismos de otimização de consumo de banda e redução da taxa de *overhead* no servidor central, contribuindo diretamente para a robustez do sistema de FL e garantido que o modelo alcance a convergência em menos rodadas de treinamento e também a redução do custo computacional. Tal abordagem contribui significati vamente para a resiliência do sistema diante do ataque de envenenamento de modelos, um dos principais e mais letais ataques relatados na literatura [Han et al. 2023].

A proposta também tem como foco a simplificação da tarefa de classificação de clientes em vista da complexa operação do FL, fornecendo um método com pouco acréscimo de processamento computacional e sem comprometer a eficácia do modelo treinado. Tendo LEGATO [Varma et al. 2021] como a principal referência para a realização deste trabalho, foi examinada a variância das normas de cada modelo, sob ataque e também na sua forma padrão, chegando à conclusão de que é possível usá-las para identificar os possíveis maliciosos. Assim, utiliza-se as normas `L1` e `L2` e também a norma `L3`, que servirá como uma generalização das outras duas normas. Este conjunto de métricas é a peça chave para o funcionamento do plugin.

A Figura 1 exemplifica como é possível identificar os dispositivos clientes com atividade suspeita, levantando um alerta para que seu gradiente seja tratado e evite o envenenamento de modelo. Cada dispositivo envia seu gradiente para a agregação logo após

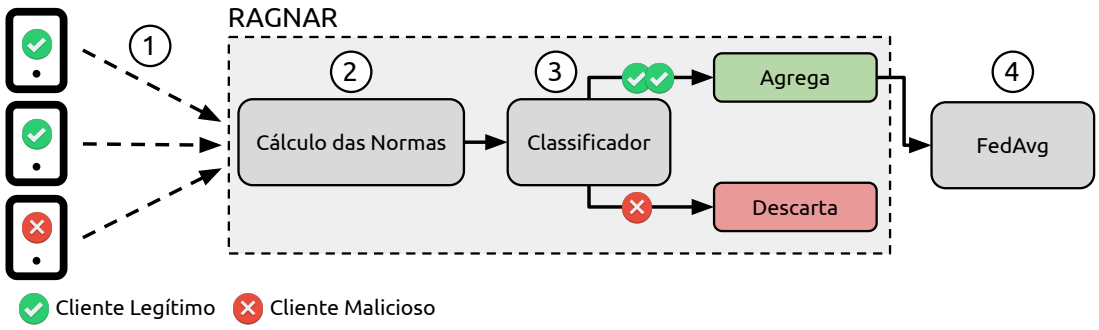


Figura 1. Diagrama de funcionamento do RAGNAR.

o treinamento (Rótulo 1, Figura 1). Aproveitando o fluxo já existente, cada gradiente é analisado antes de que os dados sejam entregues para o processo de agregação. Neste trabalho, o algoritmo de agregação considerado foi o FedAvg [McMahan et al. 2017]. O cálculo das normas é realizado a fim de mensurar a evolução do modelo a cada rodada de treinamento. Avaliando a dispersão dos vetores é possível identificar os clientes maliciosos e tratá-los adequadamente. (Rótulo 2, Figura 1). Além disso, é empregado o uso de um agente classificador para realizar o processo de identificação, assim pode-se tratar os gradientes considerados maliciosos de maneira adequada, enquanto os gradientes benignos são agregados normalmente (Rótulos 3 e 4, Figura 1). O agente classificador utilizado é um modelo *XGBoost*, visto seu alto desempenho e confiabilidade, como é mostrado em [Martin and Chai 2022].

De maneira geral, o RAGNAR é iniciado calculando três normas que serão os parâmetros de entrada para o agente classificador, sendo o cálculo das normas e o agente classificador as dependências de funcionamento do mecanismo. Deve-se destacar também que, para o funcionamento correto, os parâmetros retornados pelos clientes devem ser avaliados antes da etapa de agregação. Cada uma dessas normas distintas operam de maneira específica, essencialmente mensurando a dispersão e magnitude dos valores no vetor de cada uma das camadas do modelo. Essas métricas serão explicadas em detalhes nas subseções seguintes.

3.1. Norma `L1`

A norma `L1`, também conhecida como norma de Manhattan ou norma do valor absoluto, é um conceito importante no contexto de ML, especialmente em problemas de regularização. A norma `L1` de um vetor é a soma dos valores absolutos dos seus componentes. Em termos matemáticos, se for fornecido um vetor X com n componentes $X_1, X_2, \ldots, X_n$ a norma será dada pela Equação (1).

$$||X||_1 = |X_1| + |X_2| + \cdots + |X_n| \tag{1}$$

A norma `L1` é frequentemente utilizada como parte de técnicas de regularização, que são métodos para evitar sobre-ajuste (*overfitting*) em modelos. O *overfitting* ocorre quando um modelo se ajusta muito bem aos dados locais de treinamento, mas não generaliza bem para novos dados. A regularização `L1` é incorporada aos algoritmos de ML adicionando a norma `L1` dos pesos do modelo à função de custo durante o treinamento. A adição desta norma incentiva o modelo a ter pesos mais esparsos, ou seja, a eliminar pesos menos importantes levando seus valores a zero, contribuindo assim para a simplificação do modelo, onde se tem apenas as características mais importantes.

Um exemplo comum de regularização `L1` é chamado de Regressão *Least Absolute Shrinkage and Selection Operator (LASSO)*, onde a função de custo é ajustada adicionando a norma `L1` dos pesos multiplicada por um parâmetro de regularização à função de custo original.

3.2. Norma `L2`

A norma `L2`, ou norma Euclidiana, consiste em elevar cada termo do vetor ao quadrado, somá-los e, posteriormente, calcular a raiz quadrada dessa soma, mensurando a distância entre dois pontos utilizando a geometria euclidiana. Esta norma é dada pela Equação (2).

$$||X||_2 = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2} \tag{2}$$

Assim como a norma `L1`, a norma `L2` é frequentemente usada em técnicas de regularização para evitar *overfitting* em modelos de ML. A regularização `L2`, também conhecida como *Ridge regularization*, adiciona a norma `L2` dos parâmetros do modelo à função de custo durante o treinamento. A adição dessa norma tem o efeito de penalizar pesos grandes, incentivando o modelo a ter pesos menores e evita *overfitting*. Também é especialmente útil quando há multicolinearidade entre as características do modelo, o que significa que algumas características são altamente correlacionadas, o que não é interessante para um modelo robusto.

3.3. Norma `L3`

A norma `L3` serve como uma generalização das normas `L1` e `L2`, calculando a raiz cúbica da soma dos cubos dos elementos no vetor. Essa norma não é muito usual, porém servirá como uma validação das anteriores, já que a norma `L3` pode ser mais sensível a determinados padrões, uma vez que envolve a raiz cúbica dos valores absolutos dos pesos. Apesar de não ser tão usual, essa norma se faz necessária no contexto desta solução, visto que o seu valor diminui a medida que os *outliers* aparecem, ao contrário das demais normas. A norma `L3` pode ser representada pela Equação (3):

$$||X||_3 = \sqrt[3]{x_1^3 + x_2^3 + \ldots + x_n^3} \tag{3}$$

Como dito anteriormente, essas métricas são comumente empregadas como normalizações, porém no contexto atual, serão usadas para medir a evolução do modelo entre as rodadas de treinamento local, realizando essa verificação camada a camada e obtendo diversas métricas de um mesmo cliente, permitindo uma análise cuidadosa para classificação como anômalo ou não.

4. Avaliação de Desempenho

Esta seção descreve a metodologia e as métricas utilizadas para avaliar a eficiência do RAGNAR. Além disso, apresenta e discute os resultados obtidos.

4.1. Metodologia de avaliação

O RAGNAR foi implementado em Python no *framework* Flower [Beutel et al. 2020] e no TensorFlow, nas versões 3.10.12, 2.0.1 e 2.13.0, respectivamente. Foram considerados os *datasets* MINIST [Deng 2012] e CIFAR-10 [Krizhevsky et al. 2009], que são amplamente utilizados pela comunidade de ML.

O MNIST consiste em um conjunto de 70.000 imagens em preto e branco, com 60.000 imagens no conjunto de treinamento e 10.000 imagens no conjunto de teste. Cada imagem possui uma resolução de 28x28 *pixels* e representam dígitos escritos à mão, variando de 0 a 9. Já o CIFAR-10, conta com um conjunto de 60.000 imagens coloridas, separadas em um subconjunto de 50.000 imagens para teste e 10.000 imagens para treino. Cada uma das imagens tem resolução de 32x32 *pixels* e três canais de cores (RGB). Estas figuras estão dividas em 10 classes diferentes, onde são representadas figuras como avião, cavalo, caminhão, navio entre outras.

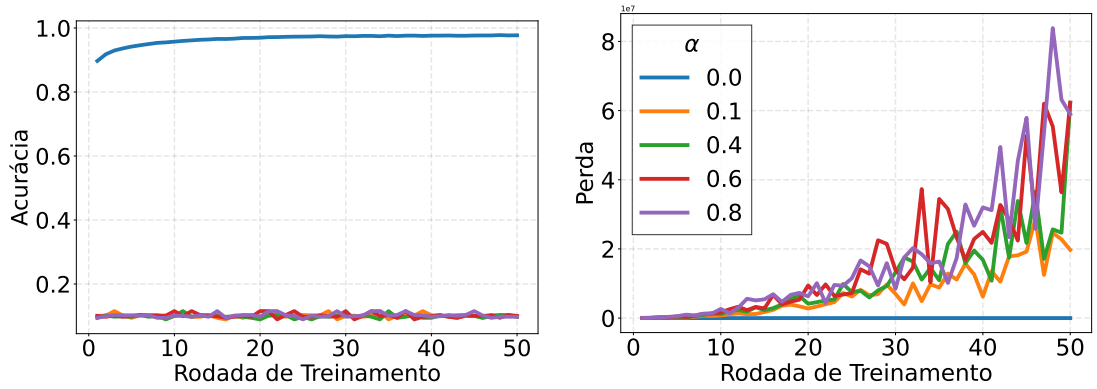
Os experimentos consideram tanto cenários com dados IID quanto não-IID. Isso é importante para confirmar a eficácia da solução em ambientes com heterogeneidade estatística dos dados dos clientes. A heterogeneidade estatística é obtida através da utilização da Distribuição de Dirichlet, variando o parâmetro α da distribuição. Quanto mais próximo de 0, mais não-IID é o cenário. Para o treinamento do agente classificador foi gerado um *dataset* sintético abrangendo os diversos exemplos de ataque com cerca de 2 milhões de linhas. As redes neurais utilizadas foram uma Rede Neural Convolucional (CNN) para os experimentos com o conjunto de dados CIFAR-10 e uma Rede Neural Densa (DNN) para o conjunto de dados MNIST. Isso se deve ao fato que assim podemos expor nossa solução a dois cenários adversos. O treinamento com o conjunto de dados CIFAR-10 é um bom desafio para validar o modelo.

Para avaliação da proposta, as seguintes métricas serão utilizadas: *(i)* **Resiliência**: Será avaliada a capacidade do modelo classificador de se ajustar aos variados ataques relatados, mantendo um bom nível de rendimento. *(ii)* **Matriz confusão**: Será utilizado uma matriz confusão para avaliar suas saídas para que seja possível identificar se o modelo realmente tem bons níveis de acerto, durante a execução do treinamento federado. Por fim, ressalta-se que neste trabalho todos os experimentos são realizados considerado 8 (40%) dos clientes como bizantinos. Uma taxa elevada quando comparada a cenários reais.

4.2. Definindo o modelo de ataque

A fim de mensurar o efeito do ataque de envenenamento de modelo, a Figura 2(a) demonstra o impacto e variação da acurácia após acréscimo de ruído Gaussiano simples. Nesta avaliação, o α utilizado varia entre 0 e 1. A avaliação dessa modalidade de ataque pode ser descrita da seguinte maneira: Os dispositivos bizantinos, após receberem os parâmetros enviados pelo servidor, realizaram o treinamento normalmente. Antes que esse valores fossem retornados, foram gerados pequenos distúrbios aleatórios, utilizando o conceito matemático da distribuição normal, onde o α foi utilizado como média da distribuição. Estes distúrbios foram somados aos valores de parâmetro e então enviados ao servidor. A Figura 2(b) mostra o efeito sobre a *perda* onde é possível notar a invalidade do modelo final com diferentes valores de α. Nessa demonstração foi avaliado o treinamento de uma DNN utilizando dataset MNIST.

Como pode ser visto, uma pequena variação nos parâmetros impossibilita que o modelo tenha um bom nível de acurácia. Esse experimento demonstra que o mecanismo padrão faz a agregação dos valores tendo como premissa que todos os participantes são legítimos e confiáveis. Evidencia também que não é necessário conhecimento avançado da federação para que obtenha sucesso em seu ataque, visto que a simples adição de valores aleatórios é suficientemente prejudicial ao treinamento.



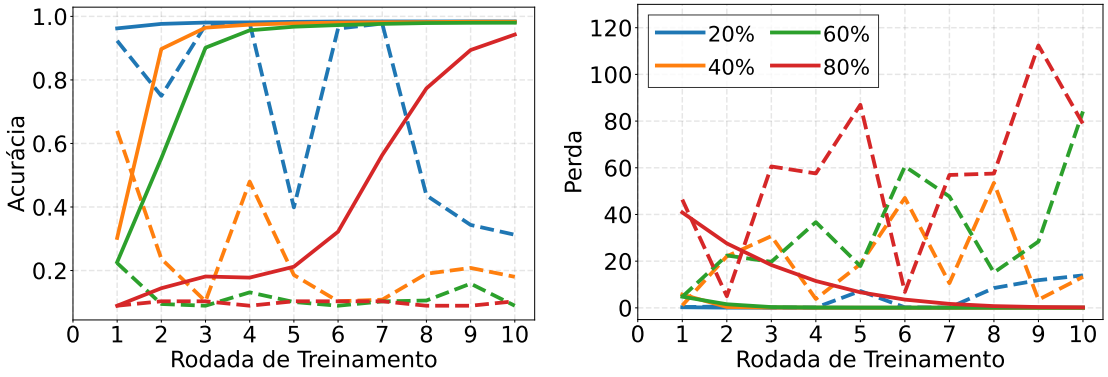
(a) Acurácia (b) Perda

Figura 2. Efeito de diferentes valores de alfa (α) do ruído gaussiano.

4.3. Eficiência do modelo de ataque

Essa avaliação é importante para mostrar como o modelo se comporta frente ao aumento do número de atacantes presentes durante o treinamento. Na Figura 3 é possível observar dois treinamentos, um destes sem RAGNAR representados pelas linhas tracejadas, e outro com RAGNAR representado pela linha contínua. É possível observar os efeitos sob diferentes proporções de atacantes. As métricas avaliadas são acurácia e perda, respectivamente. Nota-se que mesmo com o aumento de atacantes, RAGNAR mantém o processo de treinamento estável.

As seções seguintes mostram diferentes modelos de ataques e como o RAGNAR se comporta na presença dos mesmos. Essa avaliação é importante para demonstrar a eficiência da proposta frente a diferentes tipos de ataques. Foram consideradas 20 rodadas de treinamento, com variação do momento em que o ataque ocorre. Dessa forma, os ataques ocorrem nas rodadas $2, 4, 6, 8$ e 10. Isso é importante para demonstrar como o RAGNAR se comportante em diferentes estágios do processo de treinamento federado.

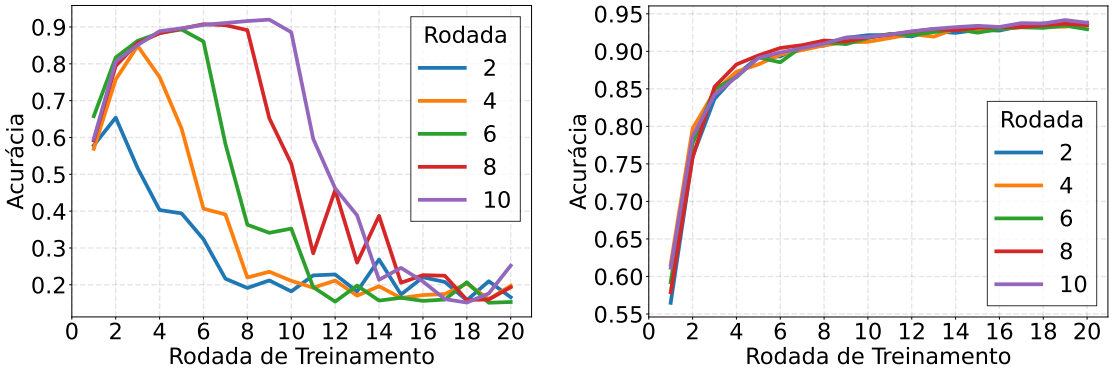


(a) Acurácia (b) Perda

Figura 3. Efeito da variação na porcentagem de clientes atacantes.

4.3.1. Ataque com zeros

Nessa modalidade, clientes bizantinos substituem os parâmetros do modelo por vetores de zeros. O objetivo deste ataque é tornar o modelo incapaz de retornar saídas válidas, visto que todas as entradas do modelo serão levadas a zero. Essa modalidade também tende a diminuir o valores de parâmetros, visto que estarão contribuindo com valores nulos. Esse ataque se mostrou pouco eficaz, visto que os *Bias* acrescentam seus valores aos pesos zerados, mantendo a saída próxima ao real. Para alcançar seu objetivo, o atacante deveria substituir todas as camadas do modelo, o que torna o ataque fácil de ser detectado.



(a) Treinamento sem RAGNAR (b) Treinamento com RAGNAR

Figura 4. Treinamento sob ataque de zeros.

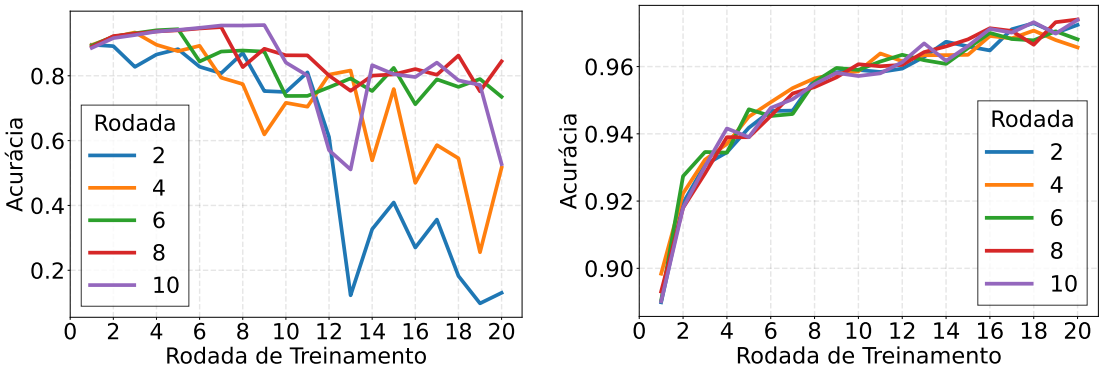
Na Figura 4(a) observa-se o treinamento de uma DNN com dataset MNIST. É possível observar o efeito desse ataque quando não se tem oRAGNAR ativado. Destaca-se que o ataque é visivelmente eficiente, reduzindo as métricas de avaliação a níveis realmente criticos no exato instante em que o ataque acontece. Por outro lado, a Figura 4(b) demonstra a eficiência da solução frente a esse ataque, nas mesmas condições em que foi exposta a federação do experimento passado, e notavelmente consegue impedir o ataque, mantendo o treinamento constante e com poucas variações de desempenho.

4.3.2. Inversão de convergência

Nessa modalidade, dispositivos malicioso geram um novo modelo local com a intenção de mudar a direção de convergência do modelo global. Isso ocorre somando o modelo global com a diferença do modelo local. Pode-se observar nas Figuras 5(a) e 5(b) o treinamento de uma DNN com dataset MNIST em configuração IID. Além disso, as Figuras 6(a) e 6(b) apresentam uma avaliação de uma rede CNN com o dataset CIFAR-10 e configuração de dados IID. Nota-se em ambos o casos o método proposto impede o ataque e mantém o treinamento como o esperado até o final do processo.

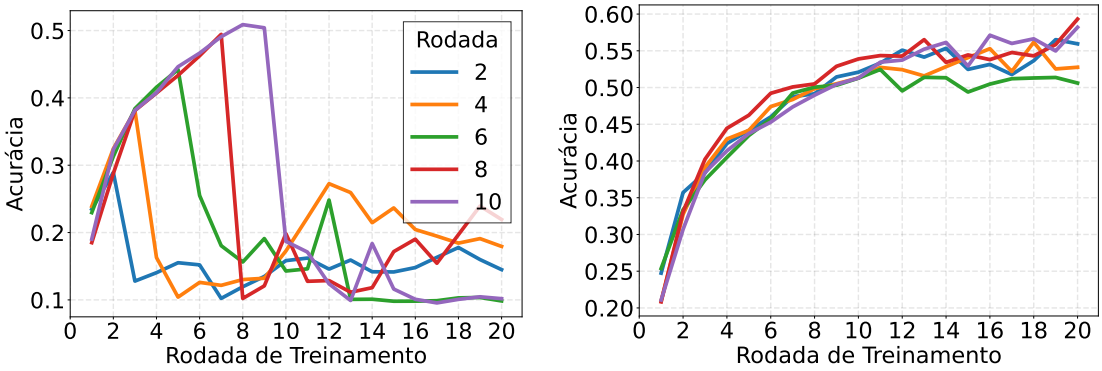
4.3.3. Inversão do gradiente

Foi avaliado também como o modelo se comporta frete a inversão do vetor de parâmetros. Observa-se que esta modalidade causa considerável dano ao modelo. Por outro lado, nota-se a eficácia do RAGNAR em defender o treinamento, mantendo a integridade do modelo



(a) Solução desativada. (b) Solução ativada.

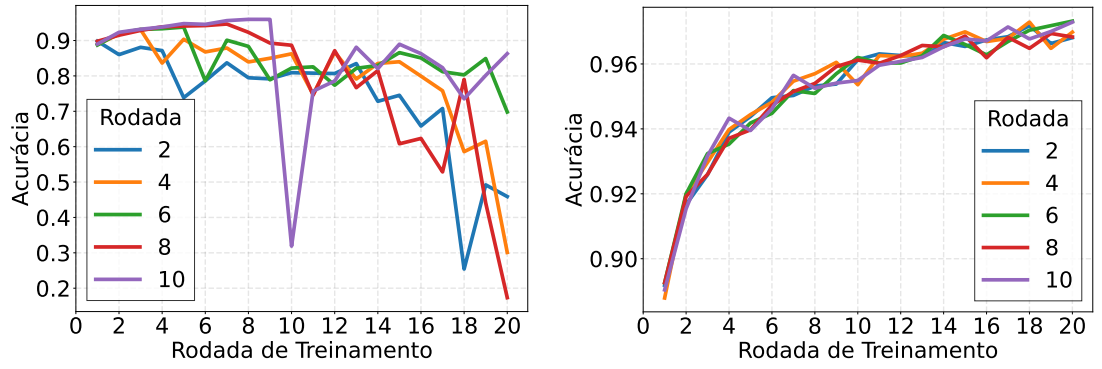
Figura 5. Eficácia do RAGNAR frente a inversão de gradiente, considerando o dataset MNIST e uma rede DNN.



(a) Solução desativada. (b) Solução ativada.

Figura 6. Eficácia do RAGNAR frente a inversão de gradiente, considerando o dataset CIFAR-10 e uma rede CNN.

e garantindo a tendência de convergência. Figuras 7(a) e 7(b) mostram o treinamento de uma DNN com dataset MNIST em configuração IID.



(a) Solução desativada. (b) Solução ativada.

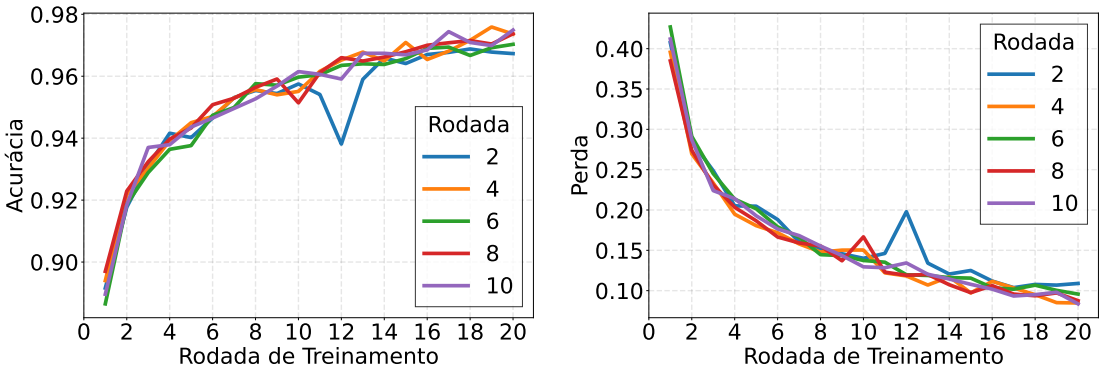
Figura 7. Eficácia do RAGNAR frente ao ataque.

É importante salientar que o modelo sofre um impacto significativo no momento da inversão do gradiente, baixando a acurácia drasticamente, mas logo em seguida se

recupera. Isso demonstra que as camadas do modelo não estão totalmente conectadas e conseguem realocar novamente para que assim seja possível funcionar corretamente. Porém, caso o atacante tenha conhecimento do número de rodadas a ser executado, poderia atacar na última rodada, invalidando o modelo completamente.

4.3.4. Ataque antes do treinamento

Nessa modalidade, o atacante realiza a inversão do vetor de parâmetros antes de que o modelo seja treinado. A intenção é que ao ser treinado o modelo mascare o ataque, visto que o dispositivo fará os ajustes necessários para que o modelo tenda a convergência. Desta forma, essa modalidade de ataque dificilmente é detectada visto que o ataque fica camuflado no treinamento local. A Figura 8 mostra os resultados com RAGNAR ativado, considerando uma rede DNN com o dataset MNIST.



(a) Acurácia. (b) Perda.

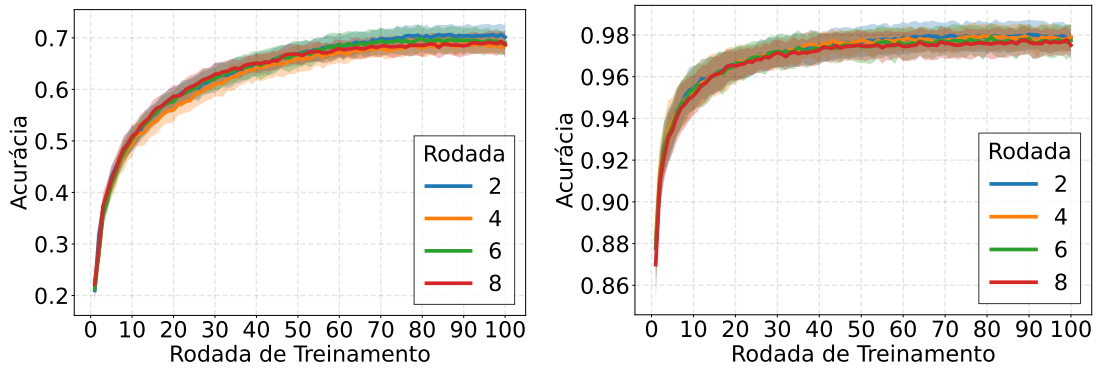
Figura 8. Eficácia do ataque e defesa.

Pode-se observar que o modelo mantém a convergência e também mantém baixos níveis de perda. Desta forma, pode-se avaliar que o RAGNAR também é uma opção de defesa contra essa modalidade.

4.4. Avaliação do modelo de defesa

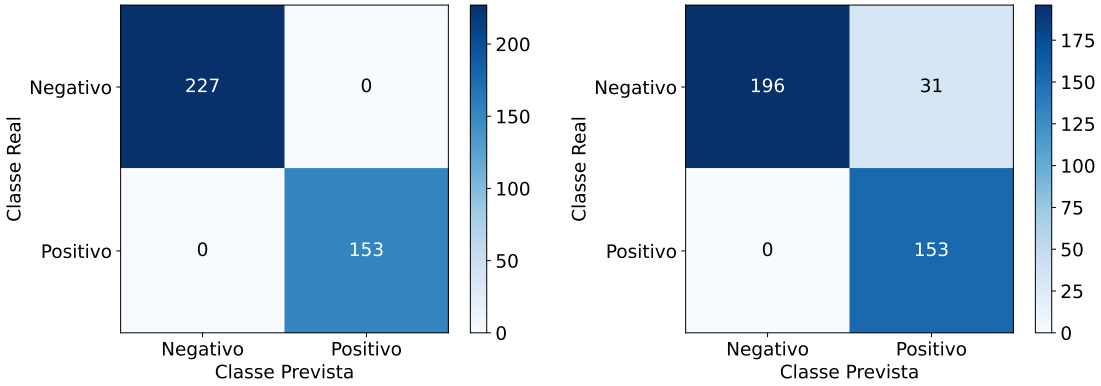
Esta seção avalia a evolução do treinamento ao longo de 100 rodadas, com ataques acontecendo em diferentes rodadas (mais especificamente nas rodadas $2, 4, 6$ e 8). O modelo comporta-se de forma estável e demonstra estar apto para realizar a tarefa proposta, demonstrando sua resiliência e robustez. Note que, o comportamento de ambos os treinamentos continua seguindo o padrão esperado para um sistema de FL sem atacantes.

Adicionalmente, a Figura 10 demonstrado a avaliação do RAGNAR utilizando a matriz confusão. Essa avaliação é importante para compreender se realmente o classificador está acertando em suas decisões. A Figura 10(a) demonstra a precisão de suas previsões em um treinamento de CNN com dataset CIFAR-10 em uma distribuição não-IID utilizando distribuição de *Dirichlet* de 0.5 sob ataque de inversão de convergência de 40% dos seus clientes participantes. É possível observar que o classificador é extremamente assertivo, identificando corretamente o ataque. É necessário destacar que o funcionamento do RAGNAR não foi afetado frente ao dataset extremamente desbalanceado, mantendo 100% de assertividade em suas classificações.



(a) CNN em 100 rounds. (b) DNN em 100 rounds.

Figura 9. Eficácia do RAGNAR ao longo do tempo de treinamento.



(a) CNN. (b) DNN.

Figura 10. Matriz de Confusão de avaliação da eficiência do RAGNAR no ataque de Inversão de Convergência.

Na Figura 10(b), observa-se que o classificador errou em alguns casos durante o treinamento de uma DNN com dataset MNIST em distribuição IID sob ataque de 40% dos seus clientes. Esse treinamento avalia o ataque de inversão de convergência. O modelo errou ao classificar clientes legítimos como atacantes, este erro é devido a modalidade de ataque avaliado. O fato de clientes legítimos serem classificados erroneamente será avaliado em estudos futuros.

5. Conclusão

Este trabalho apresentou o RAGNAR, um plugin desenvolvido para mitigar em tempo real ataques no ambiente de Aprendizado Federado. Os resultados obtidos demonstram que o método proposto é capaz de mitigar o resultado de dos principais ataques de forma substancial, gerando variações mínimas na acurácia do modelo, especialmente quando o ataque é deflagrado em rounds mais avançados do treinamento. Também foi avaliado o comportamento da acurácia do modelo quando submetido ao ataque desde o primeiro round de treinamento, com 80% dos participantes atuando como agentes atacantes. Apesar de uma redução inicial na performance, observou-se que, a partir do quinto round, a performance do modelo se recupera, e volta a evoluir de uma forma muito próxima ao observado em um ambiente isento de ataques.

Neste trabalho também foi demonstrado que os ataques ao ambiente federado são, de forma geral, eficientes e comprometem o resultado de toda a federação, reduzindo a acurácia do processo de treinamento de todos os dispositivos federados. Esses resultados corroboram o trabalho de [Sattler et al. 2020], e evidencia a necessidade de uma técnica robusta que assegure a segurança do treinamento, e que simultaneamente, seja simples e de custo computacional reduzido, visto que uma federação pode contar com centenas ou mesmo milhares de aparelhos conectados de uma só vez.

6. Agradecimentos

Este projeto foi apoiado pelo Ministério da Ciência, Tecnologia e Inovação, com recursos da Lei nº 8.248, de 23 de outubro de 1991, no âmbito do PPI-Softex, coordenado pela Softex e publicado como Agentes inteligentes para plataformas móveis baseados em tecnologia de Arquitetura Cognitiva [Processo 01245.013778/2020-21].

Referências

Abdulrahman, S., Tout, H., Ould-Slimane, H., Mourad, A., Talhi, C., and Guizani, M. (2021). A survey on federated learning: The journey from centralized to distributed on-site learning and beyond. *IEEE Internet of Things Journal*, 8(7):5476–5497.

Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Fernandez-Marques, J., Gao, Y., Sani, L., Kwing, H. L., Parcollet, T., Gusmão, P. P. d., and Lane, N. D. (2020). Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*.

Blanchard, P., El Mhamdi, E. M., Guerraoui, R., and Stainer, J. (2017a). Machine learning with adversaries: Byzantine tolerant gradient descent. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Blanchard, P., Mhamdi, E. M. E., Guerraoui, R., and Stainer, J. (2017b). Byzantine-tolerant machine learning. *CoRR*, abs/1703.02757.

Briggs, C., Fan, Z., and Andras, P. (2020). Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9.

Cho, Y. J., Wang, J., and Joshi, G. (2020). Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. *CoRR*, abs/2010.01243.

de Souza, A. M., Bittencourt, L. F., Cerqueira, E., Loureiro, A. A., and Villas, L. A. (2023). Dispositivos, eu escolho vocês: Seleção de clientes adaptativa para comunicação eficiente em aprendizado federado. In *Anais do XLI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 1–14. SBC.

de Souza, A. M., Maciel, F., da Costa, J. B. D., Bittencourt, L. F., Cerqueira, E., Loureiro, A. A., and Villas, L. A. (2024). Adaptive client selection with personalization for communication efficient federated learning. *Ad Hoc Networks*, page 103462.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142.

Fang, M., Cao, X., Jia, J., and Gong, N. (2020). Local model poisoning attacks to Byzantine-Robust federated learning. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1605–1622. USENIX Association.

Han, S., Buyukates, B., Hu, Z., Jin, H., Jin, W., Sun, L., Wang, X., Wu, W., Xie, C., Yao, Y., Zhang, K., Zhang, Q., Zhang, Y., Avestimehr, S., and He, C. (2023). Fedmlsecurity: A benchmark for attacks and defenses in federated learning and federated llms.

Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., and Li, B. (2018). Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 19–35.

Jian, X., Parthasarathy, R., and Huang, B. W. (2023). An exploratory study on the design of emergency first aid privacy protection computing system based on blockchain. In *2023 8th International Conference on Business and Industrial Research (ICBIR)*, pages 447–451.

Korkmaz, A., Alhonainy, A., and Rao, P. (2022). An evaluation of federated learning techniques for secure and privacy-preserving machine learning on medical datasets. In *2022 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, pages 1–7.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. *Toronto, ON, Canada*.

Liu, Z., Liu, Z., and Yang, X. (2023). Poisoning attack based on data feature selection in federated learning. In *2023 13th International Conference on Cloud Computing, Data Science and Engineering (Confluence)*, pages 106–110.

Martin, D. and Chai, S. S. (2022). A study on performance comparisons between knn, random forest and xgboost in prediction of landslide susceptibility in kota kinabalu, malaysia. In *2022 IEEE 13th Control and System Graduate Research Colloquium (ICSGRC)*, pages 159–164.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR.

Mehmuda, D., Bhagat, C., Patel, D., Captain, K., and Parmar, A. (2023). Defense against byzantine attack in cognitive radio using isolation forest. In *2023 15th International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pages 314–318. IEEE.

Mhamdi, E. M. E., Guerraoui, R., and Rouault, S. (2018). The hidden vulnerability of distributed learning in byzantium.

Parmar, A., Shah, K., Captain, K. M., López-Benítez, M., and Patel, J. R. (2024). Gaussian mixture model-based anomaly detection for defense against byzantine attack in cooperative spectrum sensing. *IEEE Transactions on Cognitive Communications and Networking*, 10(2):499–509.

Sattler, F., Müller, K.-R., Wiegand, T., and Samek, W. (2020). On the byzantine robustness of clustered federated learning. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8861–8865.

Varma, K., Zhou, Y., Baracaldo, N., and Anwar, A. (2021). Legato: A layerwise gradient aggregation algorithm for mitigating byzantine attacks in federated learning. In *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*, pages 272–277.

Yin, D., Chen, Y., Ramchandran, K., and Bartlett, P. (2021). Byzantine-robust distributed learning: Towards optimal statistical rates.