Algoritmo de Aprendizado de Máquina Federado Baseado em Swarm Intelligence com Privacidade Diferencial Local

Júlia Almeida Luna[1], Cauã Ferreira Sathler Siman[1], Layane Garcia Pereira[2], Thiago Lucas de Oliveira[3], Eliezer Timoteo da Silva Sanhá[4], Frederico Gadelha Guimarães[5]

[1]Ciência da Computação, Universidade Federal de Minas Gerais

[2]Matemática Computacional, Universidade Federal de Minas Gerais

[3]Sistemas de Informação, Universidade Federal de Minas Gerais

[4]Programa de Pós-Graduação em Ciência da Computação (PPGCC)
Universidade Federal de Minas Gerais (UFMG)

[5]Laboratório FutureLab, Departamento de Ciência da Computação (DCC)
Universidade Federal de Minas Gerais (UFMG)
Av. Antônio Carlos, 6627, Belo Horizonte MG, Brasil. CEP: 31270-901

fredericoguimaraes@dcc.ufmg.br

Abstract. *In federated learning, malicious agents may exploit different types of cyberattacks to manipulate the results of predictive models or to infer information about the training data. To mitigate such risks, this paper presents Fed-DP-PSO, a federated machine learning method that combines Swarm Intelligence and Local Differential Privacy. The approach aims to protect distributed data and hinder the extraction of sensitive information. The experiments conducted indicate that Fed-DP-PSO is a promising solution for training models in federated contexts with differential privacy, as its performance proved superior compared to the FedAvg method with DP-SGD.*

Resumo. *No aprendizado federado de redes neurais, agentes maliciosos podem explorar diferentes tipos de ciberataques para manipular os resultados de modelos preditivos ou para inferir sobre os dados de treinamento. Para mitigar tais riscos, este artigo apresenta o Fed-DP-PSO, um método de aprendizado de máquina federado que combina Swarm Intelligence e Privacidade Diferencial Local. A abordagem visa proteger os dados distribuídos e dificultar a extração de informações sensíveis. Os experimentos realizados indicam que o Fed-DP-PSO é promissor para o treinamento de modelos em contextos federados com privacidade diferencial, uma vez que seu resultado se mostrou superior em relação ao método FedAvg com DP-SGD.*

1. Introdução

Com o avanço da inteligência artificial nos últimos anos, impulsionado pela crescente demanda por processos automatizados, a preocupação com a confidencialidade dos modelos de redes neurais e dos dados de treinamento utilizados tornou-se cada vez mais relevante. No trabalho de (Narayanan and Shmatikov 2008), demonstra-se que, mesmo em datasets

amplamente anonimizados e com alta dimensionalidade, basta um pequeno conjunto de informações auxiliares – por exemplo, 5 a 10 atributos – para reidentificar registros individuais com alta precisão. Usando o caso do Netflix Prize, os autores evidenciam que ataques de de-anonimização podem recuperar dados sensíveis, revelando a vulnerabilidade dos dados de treinamento e ressaltando a necessidade de mecanismos robustos para protegê-los.

Além dos riscos associados à de-anonimização dos dados, a incorreta manutenção de informações pessoais por grandes empresas tem gerado diversos casos de graves violações de privacidade. Por exemplo, empresas como Quora, Google e Facebook sofreram vazamentos que atingiram mais de 100 milhões de usuários, comprometendo nomes, endereços de e-mail e senhas criptografadas (Silveira et al. 2023). Além desses casos, os hotéis Marriott tiveram os dados de 500 milhões de clientes acessados por *hackers* (Yu et al. 2022). Com isso, esses casos destacam ainda mais a importância da implementação de medidas de segurança robustas que protejam informações sensíveis contra acessos não autorizados e possíveis violações de privacidade em dados de treinamento de redes neurais.

Nesse cenário, agentes mal-intencionados podem explorar diferentes tipos de ciberataques para manipular os resultados de modelos preditivos, provocar classificações incorretas na fase de teste ou inferir informações privadas dos dados de treinamento (Jagielski et al. 2018). Para mitigar tais riscos, diversas técnicas de proteção de privacidade emergiram, entre elas o Aprendizado Federado (FL, do inglês *Federated Learning*) (Nguyen et al. 2024; Luzón et al. 2024) e a Privacidade Diferencial (DP, do inglês *Differential Privacy*) (Pan et al. 2024).

O aprendizado de máquina federado reduz a necessidade de compartilhamento direto dos dados ao aproximar a computação das fontes de informação. No entanto, embora essa abordagem fortaleça a proteção da privacidade, ela também amplia a superfície de ataque, introduzindo novos pontos vulneráveis na comunicação entre os clientes (detentores dos dados de treinamento) e o servidor central (responsável pelo modelo global). Em FL, os parâmetros do modelo treinado localmente são transmitidos ao servidor, que é assumido como confiável, ou seja, incapaz de utilizar os parâmetros recebidos para inferir informações privadas dos dados. Entretanto, tais parâmetros podem ser alvos de ataques de inferência de pertencimento e ataques de reconstrução caso sejam interceptados. Por exemplo, em (Leite et al. 2024), por meio de uma versão aprimorada do algoritmo de vazamento profundo de gradientes (DLG-FB), os autores mostram ser possível reconstruir imagens a partir apenas de inferências iniciais bem-sucedidas, tornando os ataques ainda mais assertivos. Adicionalmente, a técnica DP surge como um mecanismo de proteção adicional ao oferecer garantias matemáticas e estatísticas de privacidade. Embora sua aplicação tradicional possa limitar significativamente a utilidade dos dados de treinamento (Tatarakis 2019), estudos recentes demonstram que, quando combinada com algoritmos de otimização por enxame, a perturbação introduzida pela DP pode não apenas garantir privacidade, mas também melhorar a diversidade do enxame e evitar a convergência prematura para ótimos locais (Zhang et al. 2024). Outras técnicas de proteção, como encriptação e anonimização em nuvem, também são utilizadas, mas apresentam desafios como alto custo computacional e possíveis alterações no contexto dos dados (Silveira et al. 2023).

Diante disso, (Sanhá et al. 2024) propõe a combinação dos métodos FL, PSO (do inglês, *Particle Swarm Optimization*) e SGD (do inglês, *Stochastic Gradient Descent*) para o treinamento federado de redes neurais artificiais com dados distribuídos. Essa metodologia reduz significativamente o custo de compartilhamento de parâmetros durante o treinamento e demonstra maior acurácia em comparação a métodos federados tradicionais, como FedAvg (McMahan et al. 2017) e FedPSO (Park et al. 2021). No entanto, apesar das melhorias no desempenho e na eficiência, o modelo ainda depende de um único mecanismo de proteção de privacidade, deixando os clientes vulneráveis a ataques baseados em inferência e reconstrução de dados sensíveis, explorando os parâmetros do modelo ou analisando seus padrões de resposta.

Para abordar essa limitação, o método proposto neste trabalho, denominado Fed-DP-PSO, introduz uma camada adicional de proteção ao integrar a privacidade diferencial diretamente nos clientes, combinando-a com o PSO e o SGD em aprendizado federado. Dessa forma, os parâmetros enviados ao servidor são protegidos por privacidade diferencial, dificultando a extração de informações sensíveis e fortalecendo a segurança do sistema como um todo.

O restante deste artigo está organizado da seguinte forma: a Seção 2 aborda a fundamentação teórica, enquanto a Seção 3 descreve o método proposto. Em seguida, a Seção 4 apresenta os experimentos realizados e seus resultados. Por fim, a Seção 5 traz as conclusões do trabalho.

2. Fundamentação teórica

2.1. Aprendizado Federado

O Aprendizado Federado constitui uma abordagem descentralizada para o treinamento de modelos de aprendizado de máquina, na qual múltiplos dispositivos (também denominados usuários, clientes ou participantes) colaboram no treinamento de um modelo sem compartilhar seus dados com o servidor e uns com os outros, dessa forma aumentando o nível de privacidade para os participantes do processo. Esse paradigma surgiu como uma solução para desafios relacionados à privacidade, custos computacionais e eficiência da comunicação em sistemas distribuídos (Luzón et al. 2024).

O treinamento em FL segue um ciclo iterativo baseado em rodadas de aprendizado. O método utiliza um modelo global centralizado e um conjunto de modelos locais em cada cliente. Essa abordagem envolve N clientes $C_1, \ldots, C_N$, cada cliente C_i possui um conjunto de dados local $\mathcal{D}_i$, que não é compartilhado entre clientes ou com o servidor central. Em cada rodada, os seguintes passos são executados:

1. O servidor central distribui um modelo inicial, representado por seu vetor de parâmetros θ^t, para um subconjunto de clientes participantes, usando alguma estratégia de seleção de clientes.
2. Cada cliente atualiza seu modelo local a partir do modelo recebido pelo servidor e realiza o treinamento local em seus próprios dados, atualizando o modelo de acordo com um algoritmo de otimização.
3. As atualizações dos clientes são enviadas ao servidor, onde são agregadas para formar um novo modelo global.

O treinamento se repete até que um critério de parada seja cumprido, que pode ser um número predeterminado de iterações ou a convergência de modelo. Desde sua proposição inicial por (McMahan et al. 2017) surgiram vários trabalhos sobre FL. Destaca-se as referências a seguir para maiores detalhes e aplicações (Zhang et al. 2021; Luzón et al. 2024; Nguyen et al. 2024).

2.2. Privacidade Diferencial

O treinamento de modelos de aprendizado de máquina, especialmente redes neurais profundas, depende de grandes volumes de dados, muitas vezes sensíveis. Para garantir que informações individuais dos dados de treinamento não sejam reveladas, é necessário utilizar técnicas de proteção da privacidade. Os modelos de privacidade baseados em técnicas de anonimização de dados consideram a disponibilização de um conjunto de dados para outros atores que utilizarão esse conjunto para o treinamento de seus modelos (Majeed and Lee 2021). Contudo, um adversário ainda pode utilizar informações de fontes externas para reidentificar indivíduos, o que limita as garantias de privacidade dessas técnicas (Majeed and Lee 2021).

As técnicas de DP tornam-se importantes ferramentas para garantir a proteção da privacidade dos dados, isto é, faz alterações controladas nos resultados em forma de adição de ruídos, o que dificulta os ataques de inferência/reconstrução ao mesmo tempo afetando minimamente a utilidade dos resultados.

A premissa dessa técnica é que a presença ou ausência de um dado individual tenha um impacto limitado sobre a saída do modelo, ou seja, que os dados, individualmente, não sejam diferenciáveis. A Privacidade Diferencial é satisfeita por um algoritmo aleatório, geralmente chamado de mecanismo. Este modelo foi projetado inicialmente em um ambiente interativo, onde os usuários submetem consultas a um conjunto de dados e recebem respostas por meio de um mecanismo de anonimização.

Os conceitos fundamentais da Privacidade Diferencial relevantes para este estudo são apresentados a seguir (Pan et al. 2024).

Definição 2.1 (ε-DP)**.** Um algoritmo aleatório (mecanismo) M satisfaz ε-DP se para quaisquer dois conjuntos de dados adjacentes $\mathcal{D}$ e $\mathcal{D}'$ que diferem em no máximo um registro, e para qualquer subconjunto da saída $\mathcal{S}$ do algoritmo M, temos:

$$P[M(D) \in S] \leq e^{\varepsilon} \cdot P[M(D') \in S], \tag{1}$$

onde ε denota o orçamento de privacidade, que reflete o nível de preservação de privacidade que o algoritmo M pode fornecer. Um valor menor de ε significa proteção mais forte da privacidade.

Uma versão relaxada da privacidade diferencial pura consiste em admitir um vazamento de privacidade com uma probabilidade pequena δ.

Definição 2.2 (Privacidade diferencial aproximada)**.** Um algoritmo aleatório (mecanismo) M satisfaz (ε, δ)-DP se para quaisquer dois conjuntos de dados adjacentes $\mathcal{D}$ e $\mathcal{D}'$ que diferem em no máximo um registro, e para qualquer subconjunto da saída $\mathcal{S}$ do algoritmo M, temos:

$$P[M(D) \in S] \leq e^{\varepsilon} \cdot P[M(D') \in S] + \delta, \tag{2}$$

onde δ é um fator de relaxação.

Nessas definições, ε representa o orçamento de privacidade, que determina a quantidade de ruído adicionado às amostras, e δ é um parâmetro que permite uma pequena flexibilização da garantia de privacidade. O ajuste do δ influencia diretamente o grau de acurácia do modelo, permitindo um equilíbrio controlado entre o risco mínimo de vazamento de informação e um treinamento mais preciso.

Definição 2.3 (Mecanismo Gaussiano)**.** Seja um conjunto de dados $\mathcal{D}$ e uma função $f : \mathcal{D} \mapsto \mathbb{R}^d$. Um algoritmo aleatório do tipo mecanismo gaussiano, definido matematicamente por $M(\mathcal{D}) = f(\mathcal{D}) + \mathcal{N}(0, \sigma^2\mathbf{I})$, satisfaz (ε, δ)-DP se:

$$\sigma \geq \frac{\Delta_2}{\varepsilon}\sqrt{2\ln(1.25/\delta)} \tag{3}$$

em que:

$$\Delta_2 = \max_{\mathcal{D},\mathcal{D}'} \|f(\mathcal{D}) - f(\mathcal{D}')\|_2 \tag{4}$$

denota a sensibilidade L_2 da função para alterações de um registro no conjunto de dados. $\mathcal{N}(0, \sigma^2)$ expressa uma variável aleatória que segue a distribuição gaussiana com média 0 e desvio padrão σ.

Observe que, a partir da Definição 2.3, se a sensibilidade Δ_2 for alta, maior deverá ser o ruído adicionado para proteger a privacidade dos dados. Da mesma forma, quanto menor ε, maior deverá ser o ruído adicionado.

Na privacidade diferencial, a composição sequencial estabelece que, se um conjunto de mecanismos $M = \{M_1, \ldots, M_k\}$ for aplicado sequencialmente sobre um mesmo conjunto de dados, e cada M_i garantir ε_i-DP, então a composição resultante terá privacidade diferencial de $(\sum_{i=1}^{k} \varepsilon_i)$-DP. Isso significa que o orçamento de privacidade se acumula aditivamente, podendo degradar a proteção dos dados ao longo de múltiplas operações (Tatarakis, 2021).

Já a composição paralela ocorre quando os dados são divididos em subconjuntos disjuntos $\mathcal{D}_i$, processados por mecanismos M_i independentes. Nesse caso, a privacidade diferencial global da composição é dada por $(\max_i \varepsilon_i)$-DP, evitando o crescimento excessivo do orçamento de privacidade (Tatarakis, 2021).

Além disso, o teorema do pós-processamento garante que, se um algoritmo M_1 já satisfaz ε-DP sobre um conjunto de dados $\mathcal{D}$, qualquer processamento adicional $M_2(M_1(\mathcal{D}))$ mantém essa garantia, sem aumentar o orçamento de privacidade (Zhang, Zhu e Xie, 2024).

2.3. Otimização por Enxame de Partículas

A computação evolucionária tem se destacado como uma abordagem eficaz para a resolução de problemas complexos, em razão da simplicidade e flexibilidade dos seus algoritmos. Dentro dessa área, os métodos baseados em inteligência de enxame (*Swarm Intelligence*) são particularmente notáveis, pois se inspiram em comportamentos coletivos observados na natureza, como os de bandos de pássaros ou cardumes de peixes (Freitas et al. 2020).

Nesse contexto, (Kennedy and Eberhart 1995) desenvolveram o algoritmo Particle Swarm Optimization (PSO), que se tornou um dos principais representantes dos algoritmos de *Swarm Intelligence*. Inspirado na dinâmica de um bando de pássaros, o

algoritmo estabelece uma dinâmica complexa a partir da interação entre agentes seguindo regras simples localmente. Cada partícula se move em um espaço de busca seguindo uma equação de atualização da posição que considera informações sobre as melhores posições já alcançadas, de forma que cada partícula ajusta sua trajetória tanto com base em sua experiência individual quanto na melhor posição encontrada pelo grupo.

A dinâmica do algoritmo é formalizada por duas equações fundamentais. Seja $L(\theta) \in \mathbb{R}$ uma função de perda (*Loss function*) ou função de custo a ser minimizada no vetor de parâmetros $\theta \in \mathbb{R}^d$. A atualização da velocidade e posição de uma partícula é dada por:

$$v_i^{t+1} = \omega \cdot v_i^t + c_1 r_1 (p_{best,i}^t - \theta_i^t) + c_2 r_2 (g_{best}^t - \theta_i^t) \tag{5}$$

$$\theta_i^{t+1} = \theta_i^t + v_i^{t+1} \tag{6}$$

onde: ω é o fator de inércia das partículas, que controla a influência da velocidade anterior; v_i^t e x_i^t representam, respectivamente, os vetores da velocidade e a posição da partícula i na iteração t; c_1 e c_2 são coeficientes de aceleração, regulando a influência das componentes cognitiva (melhor posição individual) e social (melhor posição global), respectivamente; r_1 e r_2 são valores aleatórios no intervalo [0,1]; $p_{best,i}^t$ é a melhor posição individual da partícula i na iteração t, e g_{best}^t representa a melhor posição global encontrada pelo grupo na iteração t.

O ajuste cuidadoso dos hiperparâmetros c_1, c_2 e ω é fundamental para o bom desempenho do PSO, pois sua configuração determina o equilíbrio entre a exploração do espaço de busca (capacidade de varrer amplamente possíveis soluções) e a exploração intensiva em regiões promissoras (refinamento das soluções já identificadas) (Freitas et al. 2020).

3. Proposta de Treinamento Federado com Privacidade Diferencial Local

O algoritmo proposto, denominado Fed-DP-PSO, baseado no algoritmo PSO de forma colaborativa entre os clientes, busca reduzir a quantidade de comunicação de modo a reduzir a probabilidade de que informações sensíveis sejam vazadas. Ao mesmo tempo, utiliza a privacidade diferencial como uma solução na qual cada comunicação de parâmetros de modelo entre cliente-servidor é ofuscada por ruído. Se a magnitude do ruído adicionado não for muito grande, então uma boa precisão do modelo global ainda pode ser alcançada.

A Figura 1 sintetiza o fluxo de comunicação entre os clientes e o servidor no processo federado do algoritmo Fed-DP-PSO. Cada cliente transmite ao servidor o valor da função de perda associado à sua melhor partícula. O servidor identifica a partícula com o menor erro dentre todas e requisita os parâmetros correspondentes. Em seguida, esses parâmetros são definidos como a nova posição global ótima (g_{best}) e distribuídos a todos os clientes, promovendo uma otimização colaborativa orientada pelas melhores soluções locais. O processo é detalhado nas subseções seguintes, explicitando o algoritmo executado nos clientes e a atualização do modelo global no servidor. Assumimos um sistema de aprendizado federado com N clientes $C_1, \ldots, C_N$, cada cliente C_i possui um conjunto de dados local $\mathcal{D}_i$, de tal forma que

$$\mathcal{D} = \bigcup_{i=1}^{N} \mathcal{D}_i, \tag{7}$$

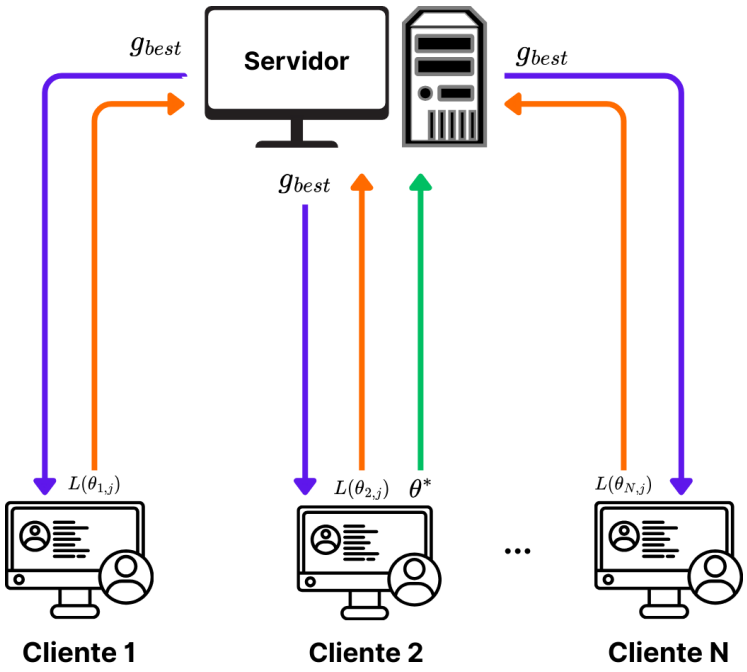

Figura 1. Algoritmo Fed-DP-PSO. Cada cliente envia seu erro $L(\theta_{i,j})$ (setas laranjas) ao servidor. O cliente escolhido entre os três com melhor desempenho envia seus parâmetros θ^* (seta verde). O servidor então atualiza o modelo global e distribui os parâmetros atualizados para os clientes (setas roxas).

em que $\mathcal{D}_i = \{x_1, \ldots, x_{m_i}\}$ é uma coleção de amostras ou observações de dados em C_i e $m_i = |\mathcal{D}_i|$. O objetivo é minimizar uma função global:

$$\min_{\theta \in \mathbb{R}^d} L(\mathcal{D}, \theta) \tag{8}$$

em um ambiente distribuído no qual muitos clientes têm seus próprios conjuntos de dados locais e o problema de minimização de soma finita é sobre a união de todos os conjuntos de dados locais.

3.1. Treinamento local

Cada cliente executa um método de treinamento híbrido combinando o algoritmo PSO com o método de descida do gradiente estocástico (SGD), mais especificamente o algoritmo Adam, usando mini-lotes do seu respectivo conjunto de dados $\mathcal{D}_i$.

Além disso, para implementar DP no processo de treinamento local, adicionamos uma perturbação aleatória no gradiente calculado pelo método Adam, definindo dessa forma um mecanismo Gaussiano durante a otimização de parâmetros.

Seja $L(\mathcal{D}_i, \theta^t)$ a função de perda no treinamento do modelo local. Considerando uma amostra $x_j \in \mathcal{D}_i$, temos um estimador estocástico do gradiente da função de perda $\nabla L(\mathcal{D}_i, \theta^t)$ como sendo dado por $\nabla L(x_j, \theta^t)$ ou sua versão de mini-lote dada por:

$$\nabla L(\mathcal{B}_k, \theta^t) = \frac{1}{|\mathcal{B}_k|} \sum_{x_i \in \mathcal{B}_k} \nabla L(x_i, \theta^t) \tag{9}$$

em que $\mathcal{B}_k \subset \mathcal{D}_i$ é um mini-lote (*mini-batch*) do conjunto de dados no cliente.

A atualização dos parâmetros por meio do algoritmo Adam se dá por:

$$\theta^{t+1} = \theta^t - \eta \left(\lambda\theta^t + \nabla L(\mathbf{b}_k, \theta^t)\right) \tag{10}$$

onde η é a taxa de aprendizado e λ é um parâmetro relacionado aos momentos do gradiente.

Cada cliente utiliza um conjunto de partículas $\Theta_i = \{\theta_{i,j}\}$, em que $\theta_{i,j} \in \mathbb{R}^d$ representa uma partícula j no cliente C_i. A atualização da velocidade e posição de cada partícula é uma combinação das equações do PSO com a atualização do gradiente dado pelo método Adam:

$$v_j^{t+1} = \omega \cdot v_j^t + c_1 r_1 (p_{best,j}^t - \theta_{i,j}^t) + c_2 r_2 (g_{best}^t - \theta_{i,j}^t) \tag{11}$$

$$\theta_{i,j}^{t+1} = \theta_{i,j}^t + v_j^{t+1} - \eta \left(\lambda\theta_{i,j}^t + \nabla L(\mathcal{B}_k, \theta_{i,j}^t)\right) \tag{12}$$

Durante a otimização com Adam, a biblioteca Opacus[1] aplica três mecanismos principais para garantir a privacidade diferencial: clipagem dos gradientes, adição de ruído gaussiano e rastreio do orçamento de privacidade. Primeiro, a clipagem limita a norma dos gradientes, impedindo que contribuições individuais tenham um impacto desproporcional na atualização do modelo. Em seguida, o ruído gaussiano é adicionado aos gradientes para mascarar informações sensíveis. Por fim, um contador de privacidade baseado em Renyi Differential Privacy (RDP) rastreia o orçamento de privacidade ao longo das iterações, garantindo que o modelo permaneça dentro dos limites de privacidade estabelecidos.

O Algoritmo 1 descreve o procedimento de treinamento local realizado por cada cliente em um cenário de aprendizado federado com otimização baseada em enxame de partículas (PSO). Na primeira rodada (linha 1), é realizada a inicialização aleatória do conjunto de partículas Θ_i no cliente i. As linhas 3 a 7 correspondem à fase de avaliação local, na qual cada partícula tem sua função de perda $L(\mathcal{D}_i, \theta_{i,j})$ computada. A melhor posição individual ($p_{best,j}$) é atualizada sempre que uma nova solução apresenta desempenho superior. O menor erro local entre todas as partículas, $L(\mathcal{D}_i, \theta_{i,best})$, é então enviado ao servidor central. Caso o servidor solicite, o modelo correspondente a essa solução é transmitido; caso contrário, o cliente aguarda o recebimento da melhor posição global (g_{best}). A partir da linha 16, com g_{best} definido, realiza-se a atualização da velocidade e da posição de cada partícula, conforme a dinâmica clássica do PSO, incorporando os componentes inerciais, cognitivos e sociais. Por fim (linhas 28 a 32), o modelo associado à melhor partícula local é refinado via otimização com o algoritmo Adam incluindo privacidade diferencial. Durante este refinamento, os gradientes são calculados em mini-batches, submetidos ao *clipping* e à adição de ruído gaussiano para garantir privacidade diferencial. A atualização dos parâmetros ocorre então por meio de retropropagação estocástica, assegurando tanto eficiência quanto preservação da privacidade dos dados locais.

3.2. Atualização do modelo global no servidor

Em cada rodada de comunicação, todos os clientes executam o Algoritmo 1. Em seguida, o servidor classifica os erros $L(\theta)$ dos N clientes em ordem crescente e seleciona os três

[1] `https://opacus.ai`

Data: Conjunto de dados local $\mathcal{D}_i$, número de partículas no *swarm* S.
if $round = 1$ **then**
  Crie e inicialize um swarm $\Theta_i = \{\theta_{i,j}, \ldots, \theta_{i,S}\}$;
  foreach *partícula* $j = 1, \ldots, S$ **do**
    Avalie $L(\mathcal{D}_i, \theta_{i,j})$;
    `// Atualize a melhor posição da partícula`
    if $L(\mathcal{D}_i, \theta_{i,j}) < L(\mathcal{D}_i, p_{best,j})$ **then** $p_{best,j} = \theta_{i,j}$;
  end
end
Envie o menor erro $L(\mathcal{D}_i, \theta_{i,best})$ para o servidor;
if *servidor solicita modelo local* **then**
  `// Envie os parâmetros da melhor partícula`
  Envie $\theta_{i,best}$ para o servidor;
else
  Recebe g_{best} do servidor;
end
foreach *partícula* $j = 1, \ldots, S$ **do**
  $r_1, r_2 \sim U(0, 1)$;
  `// Atualize a velocidade das partículas`
  $v_j^{t+1} = \omega \cdot v_j^t + c_1 r_1 (p_{best,j}^t - \theta_{i,j}^t) + c_2 r_2 (g_{best}^t - \theta_{i,j}^t)$;
  `// Atualize a posição das partículas`
  $\theta_{i,j}^{t+1} = \theta_{i,j}^t + v_j^{t+1}$;
end
foreach *partícula* $j = 1, \ldots, S$ **do**
  Avalie $L(\mathcal{D}_i, \theta_{i,j})$;
  if $L(\mathcal{D}_i, \theta_{i,j}) < L(\mathcal{D}_i, p_{best,j})$ **then** $p_{best,j} = \theta_{i,j}$;
end
`// Refine a melhor partícula com DP-Adam:`
for $e = 1, \ldots, E$ **do**
  foreach *mini-batch* $\mathcal{B}_k$ **do**
    Calcule $\nabla L(\mathcal{B}_k, \theta_{i,best}^t)$;
    $\theta_{i,best}^{t+1} = \theta_{i,best}^t - \eta \left\{ \lambda \theta_{i,best}^t + \left[\nabla L(\mathcal{B}_k, \theta_{i,best}^t) \right] + \mathcal{N}(0, \sigma \mathbf{I}) \right\}$;
  end
end

Algorithm 1: Algoritmo DP-PSO-Adam.

menores valores. Dentre esses três melhores clientes, um é escolhido aleatoriamente, conforme definido nas Equações 13 e 14:

$$L_{\theta_{(1)}} \leq L_{\theta_{(2)}} \leq L_{\theta_{(3)}} \leq \cdots \leq L_{\theta_{(N)}} \tag{13}$$

$$L_{\min} = L_{\theta_{(k)}} \tag{14}$$

onde $k \in \{1, 2, 3\}$ é escolhido aleatoriamente.

Essa seleção aleatória dos três melhores clientes garante que a atualização do modelo global seja diversificada, reduzindo o impacto excessivo de qualquer cliente individual na atualização do modelo. Após a escolha do $L_{\min}$, o servidor solicita os parâmetros

$\theta^*_{(k)}$ do cliente correspondente, que os envia para o servidor.

Por fim, o modelo global é atualizado usando $g_{best} = \theta^*_{(k)}$, que por sua vez é distribuído a todos os clientes, garantindo a continuidade do treinamento colaborativo. Veja que o novo g_{best} enviado pelo servidor aos clientes influenciará o treinamento local em todos os clientes, por meio das equações de atualização das partículas no cliente. Isso faz com que o treinamento distribuído em todos os clientes esteja de alguma forma conectado e direcionado por aqueles que possuem os menores erros de treinamento. A dinâmica dessa comunicação entre clientes e servidor está ilustrada na Figura 1.

O Algoritmo 2 descreve a lógica executada pelo servidor central durante o treinamento no Fed-DP-PSO. A cada rodada de comunicação, o servidor solicita a todos os clientes o menor valor de perda $L(\theta)$ associado às melhores posições pessoais encontradas localmente (linhas 2 a 4). Em seguida, esses valores são ordenados de forma crescente, e o servidor seleciona aleatoriamente um dos três clientes com os menores erros (linhas 5 a 7). O modelo correspondente, identificado por seus parâmetros θ^*, é requisitado ao cliente selecionado (linha 8). Por fim, o parâmetro global ótimo g_{best} é atualizado com θ^* e transmitido a todos os clientes, assegurando a continuidade e a coesão do processo de otimização colaborativa (linha 9).

```
1  for round = 1, ..., R do
2      for cada cliente C_i do
3          Solicita menor L_θ = L(D_i, θ) do cliente C_i;
4      end
5      Ordena os erros L_θ dos N clientes
6      Escolhe aleatoriamente k ∈ {1, 2, 3} ;
7      L_min = L_θ(k) ;
8      θ* = solicita parâmetros θ_(k) do cliente escolhido;
9      Atualiza o g_best de todos os clientes com θ*;
10 end
```

Algorithm 2: Algoritmo executado no servidor.

4. Experimentos Computacionais

4.1. Dados e configuração dos modelos

Os experimentos realizados para a validação do algoritmo Fed-DP-PSO proposto foram conduzidos utilizando três conjuntos de dados distintos: MNIST[2], Fashion MNIST[3] e PathMNIST[4]. O conjunto MNIST é composto por 60.000 imagens de treinamento e 10.000 imagens de teste, representando dígitos manuscritos entre 0 e 9, com resolução de 28×28 pixels em escala de cinza. O Fashion MNIST possui a mesma estrutura do MNIST em relação à quantidade de amostras e resolução das imagens, mas com classes que representam peças de vestuário, como camisetas, calçados e bolsas. Por fim, o PathMNIST, da base de dados MedMNIST, contém 107.180 imagens coloridas de lâminas histopatológicas do cólon, com resolução de 28×28 pixels e classificadas em 9 categorias

[2]`https://www.kaggle.com/datasets/hojjatk/mnist-dataset`

[3]`https://www.kaggle.com/datasets/zalando-research/fashionmnist`

[4]Disponível em `https://medmnist.com`

distintas. As imagens estão divididas em 89.996 para treinamento, 10.004 para validação e 7.180 para teste.

A arquitetura utilizada para os conjuntos MNIST e Fashion MNIST foi uma rede *Multi-Layer Perceptron (MLP)* composta por três camadas totalmente conectadas, sendo duas camadas ocultas com 256 neurônios cada e uma camada de saída com 10 neurônios. A função de ativação *ReLU* foi aplicada após cada camada oculta, e a camada de saída utiliza a função *softmax*. O objetivo do modelo era classificar corretamente o maior número possível de imagens nas dez classes disponíveis, assegurando simultaneamente a **privacidade e integridade dos dados** de cada cliente.

Devido à sua maior complexidade em relação aos demais *datasets*, apenas o conjunto PathMNIST foi processado utilizando uma arquitetura baseada em *Convolutional Neural Network (CNN)*, especificamente em uma variação da LeNet-5. A arquitetura utilizada processa imagens coloridas (3 canais) e é composta por duas camadas convolucionais seguidas de operações de *pooling* e três camadas totalmente conectadas. As camadas convolucionais utilizam filtros de tamanho 5×5, com 6 e 16 canais de saída, respectivamente. Após o segundo *pooling*, os mapas de ativação são achatados e processados pelas camadas lineares com 120, 84 e, por fim, 9 neurônios na camada de saída, correspondendo ao número de classes do PathMNIST. A função de ativação *ReLU* é utilizada entre as camadas convolucionais e ocultas. Neste caso, o objetivo do modelo era identificar corretamente os diferentes tipos de tecidos presentes em imagens histopatológicas, ao mesmo tempo que promovia a **preservação da privacidade** dos dados de treinamento.

Diferentes configurações de hiperparâmetros foram adotadas para os treinamentos com MLP e CNN, definidas empiricamente a partir do melhor desempenho observado em cada cenário. A principal diferença entre as arquiteturas reside na taxa de aprendizado, que foi reduzida na CNN devido à sua maior profundidade e complexidade estrutural. Essa escolha visa promover atualizações mais estáveis durante a otimização, contribuindo para uma convergência mais precisa e menos suscetível a oscilações.

Tais ajustes possibilitaram atenuar parcialmente a degradação de desempenho e aprimorar a eficiência do modelo frente às restrições impostas, especialmente nos cenários com aplicação de privacidade diferencial.

Hiperparâmetros para MLP:

- Número de épocas: 10
- Número de partículas (PSO) em cada cliente: 5
- Inércia, c_1 e c_2: 0.7, 1.4, 1.4
- Taxa de aprendizado η: 0.005
- Número de clientes N: 5

Hiperparâmetros para CNN:

- Número de épocas: 10
- Número de partículas (PSO) em cada cliente: 5
- Inércia, c_1 e c_2: 0.7, 1.4, 1.4
- Taxa de aprendizado η: 0.001
- Número de clientes N: 5

4.2. Resultados e Discussões

A apresentação e discussão dos resultados estão organizadas em três subseções principais. A Seção 4.2.1 aborda os experimentos conduzidos com o algoritmo *sem a aplicação de privacidade diferencial (DP)*. A Seção 4.2.2 trata dos experimentos realizados sob o mesmo protocolo, porém com a *inclusão de mecanismos de privacidade diferencial*. Por fim, a Seção 4.2.3 apresenta a análise comparativa do algoritmo com o Método FedAvg-DP-SGD.

Para garantir uma avaliação robusta da abordagem proposta, os experimentos foram realizados com três *datasets* distintos: MNIST, FashionMNIST e PathMNIST. Os dois primeiros foram utilizados nas seções 4.2.1 e 4.2.2 e o último foi abordado na seção 4.2.3. Além disso, é importante destacar que os dois primeiros conjuntos de dados foram utilizados com uma arquitetura MLP, enquanto o último utilizou uma CNN baseada na LeNet-5, devido à maior complexidade das imagens presentes nesse conjunto.

A distribuição dos dados entre os clientes foi realizada de duas formas: IID (do inglês, *Independent and Identically Distributed*), representando uma divisão uniforme e homogênea, e não-IID, modelada com a estratégia Dirichlet. Para tal, foram adotados os particionadores *IID Partitioner* e *Dirichlet Partitioner*, disponibilizados pela biblioteca Flower[5], especializada em aprendizado federado.

No particionamento IID, cada cliente recebe subconjuntos de dados com distribuição semelhante à global, o que favorece a convergência e o desempenho dos algoritmos. Já no particionamento Dirichlet, a distribuição dos dados é controlada por uma variável contínua $\alpha > 0$, que atua como parâmetro de concentração da distribuição Dirichlet(α) utilizada para amostrar a proporção de rótulos que cada cliente recebe. Valores menores de α resultam em vetores de proporção mais esparsos, concentrando os dados de cada cliente em poucas classes, o que induz maior heterogeneidade entre os clientes. Por outro lado, valores maiores de α geram distribuições mais balanceadas e próximas do regime IID, pois cada cliente tende a receber dados de todas as classes em proporções similares.

4.2.1. Cenário base: resultados obtidos com a ausência de privacidade diferencial

Nesta seção, são descritos os resultados no cenário base com os conjuntos de dados MNIST e FashionMNIST, que foram utilizados tanto em distribuição IID, quanto em não-IID.

O algoritmo proposto obteve as seguintes acurácias médias:

- Utilizando o *dataset* MNIST:
  - **88,69% ± 3,53 pp**, com distribuição desbalanceada e $\alpha = 0.3$;
  - **94,85% ± 0,91 pp** com distribuição desbalanceada e $\alpha = 1$;
  - **96,41% ± 0,23 pp**, com distribuição balanceada.
- Utilizando o *dataset* FashionMNIST:
  - **69,82% ± 7,53 pp**, com distribuição desbalanceada e $\alpha = 0.3$;
  - **81,86% ± 1,70 pp** com distribuição desbalanceada e $\alpha = 1$;

[5]Flower A Friendly Federated AI Framework: `https://flower.ai`

– **85,66% $\pm$ 0,99 pp**, com distribuição balanceada.

A queda de desempenho observada entre os cenários balanceado e desbalanceados era esperada, dado que a *disponibilidade limitada de certas classes em alguns clientes* compromete a representatividade local e, consequentemente, a qualidade do aprendizado federado. No entanto, é notável que a diferença entre os resultados não é acentuada, mesmo nos casos de maior assimetria de dados (quando α é menor), o que demonstra a **robustez e adaptabilidade do algoritmo proposto em ambientes não homogêneos**.

O PSO, ao operar de maneira colaborativa e global, contribui para mitigar os efeitos adversos da propriedade não-IID dos dados, favorecendo a convergência do modelo global.

4.2.2. Análise dos Resultados com Privacidade Diferencial

Nesta subseção, apresentamos os resultados obtidos sob o algoritmo (ε, δ)-DP desenvolvido, onde definimos $\delta = 10^{-5}$ e variamos $\varepsilon \in \{5,\ 8,\ 10, \infty\}$ e $\alpha \in \{0.3, 1\}$, a fim de verificarmos a eficiência do método proposto em regimes distintos de privacidade e de distribuição de dados e quantificar o trade-off entre a privacidade e a utilidade do modelo.

Foram aplicados os experimentos tanto ao treinamento com dados balanceados quanto ao treinamento com dados desbalanceados, com o intuito de não apenas verificar a eficiência do modelo, mas também sua resiliência frente a distribuições de dados realistas, onde o desbalanceamento é frequente. As Tabelas 1 e 2 apresentam os resultados da variação dos valores de ε e α no treinamento com dados balanceados e com dados desbalanceados para os *datasets* MNIST e FashionMNIST, respectivamente.

Tabela 1. Resultados de acurácia para diferentes valores de ε em treinamento com dados balanceados e desbalanceados do *dataset* MNIST.

ε	Dados não IID		Dados IID
	$\alpha = 0{,}3$	$\alpha = 1$	
5	72,31% ± 5,78 pp	85,08% ± 1,12 pp	88,12% ± 0,78 pp
8	74,61% ± 5,12 pp	86,54% ± 1,62 pp	89,21% ± 0,33 pp
10	76,24% ± 3,97 pp	87,89% ± 1,09 pp	89,59% ± 0,58 pp
∞	88,69% ± 3,53 pp	94,85% ± 0,91 pp	96,41% ± 0,23 pp

Tabela 2. Resultados de acurácia para diferentes valores de ε em treinamento com dados balanceados e desbalanceados do *dataset* Fashion MNIST.

ε	Dados não IID		Dados IID
	$\alpha = 0{,}3$	$\alpha = 1$	
5	58,92% ± 6,6 pp	76,12% ± 1,19 pp	81,34% ± 0,41 pp
8	61,18% ± 7,58 pp	76,93% ± 1,22 pp	81,83% ± 0,35 pp
10	62,06% ± 6,73 pp	77,20% ± 1,24 pp	82,06% ± 0,43 pp
∞	69,82% ± 7,53 pp	81,86% ± 1,70 pp	85,66% ± 0,99 pp

A introdução da privacidade diferencial no algoritmo proposto teve impacto direto sobre os resultados de acurácia, conforme esperado, em decorrência do ruído introduzido nas atualizações dos modelos locais.

No *dataset* MNIST, observou-se uma queda moderada na acurácia com dados não IID, em relação aos dados IID. Além disso, conforme apresentado na Figura 2, nota-se ainda que, em contextos IID, a acurácia atinge valores próximos do máximo já nas primeiras quatro rodadas de treinamento.

A mesma tendência é observada no FashionMNIST, com resultados que seguem um comportamento coerente, embora apresentem acurácias, em média, inferiores, tendo em vista a maior complexidade desse conjunto de dados em relação ao anterior.

Verifica-se também que, à medida que o valor de ε diminui, observa-se uma tendência de redução na acurácia do modelo, o que demonstra a existência de um compromisso entre a preservação da privacidade e o desempenho preditivo. Ademais, no cenário não-IID, a acurácia decresce com a redução de α, devido ao aumento da desigualdade entre os dados locais, enquanto um valor maior de α resulta em acurácia mais próxima do cenário IID, bem como esperado.

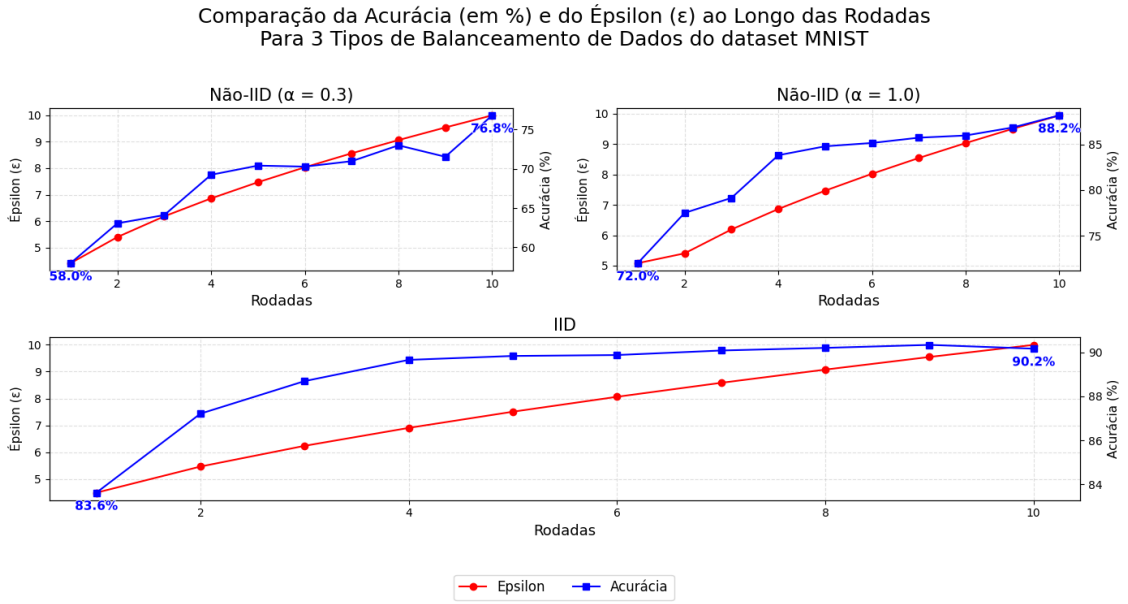

Figura 2. Comparação entre a acurácia e o épsilon (ε) ao longo das rodadas.

4.2.3. Análise Comparativa com o Método FedAvg-DP-SGD

Além dos experimentos anteriores, também avaliamos os resultados obtidos com a aplicação do método DP-SGD (Abadi et al. 2016) no treinamento local, em conjunto com a estratégia de agregação Federated Averaging (FedAvg) (McMahan et al. 2017) no servidor central. Essa abordagem foi utilizada como base comparativa para os resultados obtidos com o método proposto Fed-DP-PSO, permitindo uma análise mais abrangente do desempenho e da eficácia do algoritmo quando aplicado ao *dataset* PathMNIST.

O gráfico da Figura 3 apresenta uma comparação dos resultados obtidos com o método Fed-DP-PSO e o FedAvg-DP-SGD:

Os resultados experimentais indicam que o algoritmo proposto, Fed-DP-PSO, apresentou desempenho equivalente ao método tradicional FedAvg utilizando DP-SGD

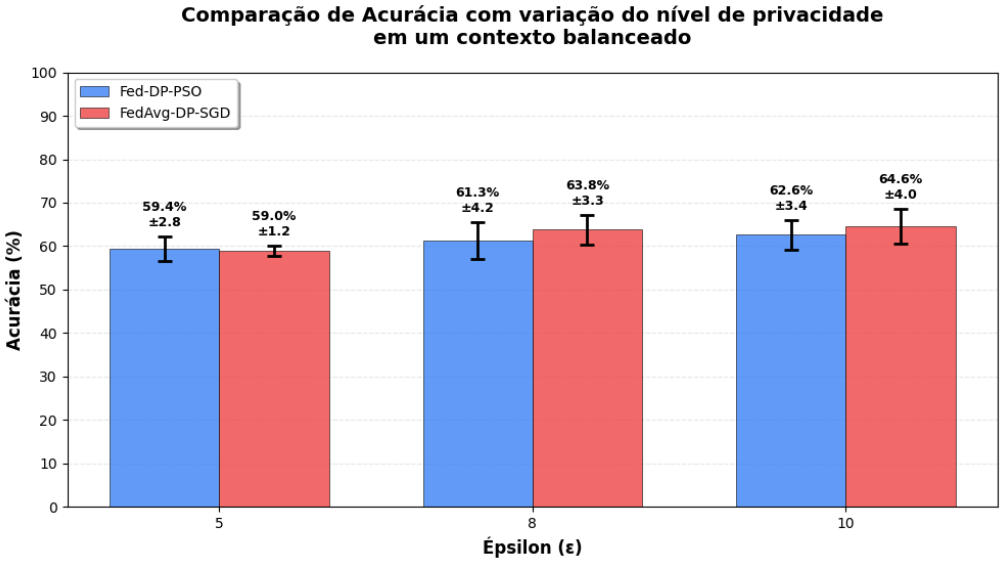

Figura 3. Comparação das acurácias obtidas com os métodos Fed-DP-PSO e FedAvg-DP-SGD com o *dataset* PathMNIST.

em todos os cenários analisados. Além disso, de acordo com (Sanhá 2024), algoritmos de aprendizado federado baseados em PSO reduzem em aproximadamente 90% a quantidade de dados transmitidos dos clientes para o servidor por rodada de treinamento, quando comparado ao FedAvg, no contexto de redes neurais convolucionais (CNNs). Isso representa uma vantagem significativa em termos de economia de recursos computacionais e segurança, uma vez que a redução de parâmetros compartilhados com o servidor diminui a superfície de exposição a ataques de *membership inference* e outros ataques de privacidade.

5. Conclusão

Este artigo propôs o método Fed-DP-PSO, que combina o treinamento federado com a aplicação de privacidade diferencial em enxames de partículas, todos alocados nos próprios clientes. Os testes foram realizados utilizando os conjuntos de dados MNIST, FashionMNIST e PathMNIST, considerando cenários de distribuição balanceada e desbalanceada, com o objetivo de simular tanto condições ideais quanto realistas, em que a colaboração entre os clientes é essencial para alcançar boa cobertura do espaço amostral. A combinação das abordagens resultou em desempenho satisfatório, com acurácia muito próxima à obtida por métodos como o FedAvg-DP-SGD, evidenciando a eficácia do Fed-DP-PSO em contextos de treinamento distribuído com proteção à privacidade. Destaca-se, ainda, a redução nos custos de comunicação e na frequência de troca de informações entre clientes e servidor, fatores que favorecem tanto a eficiência do modelo quanto o reforço à proteção dos dados. Além disso, observou-se que o uso da privacidade diferencial integrada ao PSO não apenas fortalece a proteção dos dados, mas também favorece a diversidade e a robustez do processo de otimização.

A abordagem proposta neste trabalho abre espaço para a identificação de melhorias e para a exploração de novos contextos de aplicação. Em particular, destaca-se como limitação a dificuldade de realizar múltiplos ciclos de refinamento com o algoritmo Adam,

devido ao acúmulo do orçamento de privacidade ao longo das iterações, o que impõe restrições ao número de épocas viáveis no cenário com privacidade diferencial. Entre os trabalhos futuros, pretende-se investigar alternativas para incorporar DP de forma mais eficiente, buscando reduzir o custo computacional sem comprometer a segurança, além de investigar estratégias para aprimorar a acurácia do modelo.

Agradecimentos

Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Processo 304856/2025-8, "*Aprendizado de Máquina Colaborativo e Proteção à Privacidade*"; Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) por meio do Programa de Excelência Acadêmica (PROEX) e Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG).

Parcialmente financiado por Instituto Kunumi e Embrapii, projeto PDCC-2412.0030. Parcialmente financiado pelo acordo de PDI entre a UFMG e a Fundep – Fundação de Desenvolvimento da Pesquisa: "*Programa Federado de Aprendizado de Máquina para Veículos Conectados*".

Referências

[Abadi et al. 2016] Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS'16. ACM.

[Freitas et al. 2020] Freitas, D., Lopes, L. G., and Morgado-Dias, F. (2020). Particle swarm optimisation: A historical review up to the current developments. *Entropy*, 22(3).

[Jagielski et al. 2018] 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.

[Kennedy and Eberhart 1995] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4.

[Leite et al. 2024] Leite, L., Santo, Y., Dalmazo, B., and Riker, A. (2024). Federated learning under attack: Improving gradient inversion for batch of images. In *Anais do XXIV Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, pages 794–800, Porto Alegre, RS, Brasil. SBC.

[Luzón et al. 2024] Luzón, M. V., Rodríguez-Barroso, N., Argente-Garrido, A., Jiménez-López, D., Moyano, J. M., Del Ser, J., Ding, W., and Herrera, F. (2024). A tutorial on federated learning from theory to practice: Foundations, software frameworks, exemplary use cases, and selected trends. *IEEE/CAA Journal of Automatica Sinica*, 11(4):824–850.

[Majeed and Lee 2021] Majeed, A. and Lee, S. (2021). Anonymization techniques for privacy preserving data publishing: A comprehensive survey. *IEEE Access*, 9:8512–8545.

[McMahan et al. 2017] 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.

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

[Nguyen et al. 2024] Nguyen, L. M., Hoang, T. N., and Chen, P.-Y. (2024). *Federated Learning: Theory and Practice*. Academic Press.

[Pan et al. 2024] Pan, K., Ong, Y.-S., Gong, M., Li, H., Qin, A., and Gao, Y. (2024). Differential privacy in deep learning: A literature survey. *Neurocomputing*, 589:127663.

[Park et al. 2021] Park, S., Suh, Y., and Lee, J. (2021). Fedpso: Federated learning using particle swarm optimization to reduce communication costs. *Sensors*, 21(2).

[Sanhá et al. 2024] Sanhá, E. T., Erazo-Costa, F. J., and Guimaraes, F. G. (2024). Algoritmo hıbrido de otimizaçao por enxame de partıculas para o aprendizado federado de redes neurais artificiais. In *Anais do Congresso Brasileiro de Automática (CBA)*, pages 1–6.

[Sanhá 2024] Sanhá, E. (2024). Algoritmo híbrido de otimização por enxame de partículas para o aprendizado federado de redes neurais artificiais. Master's thesis, Escola de Engenharia.

[Silveira et al. 2023] Silveira, M., Portela, A., Souza, M., Silva, D., Mesquita, M., Silva, D., Menezes, R., and Gomes, R. (2023). Aplicação de técnicas de encriptação e anonimização em nuvem para proteção de dados. In *Anais do XXIII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, pages 111–124, Porto Alegre, RS, Brasil. SBC.

[Tatarakis 2019] Tatarakis, N. (2019). Differentially private federated learning. Master's thesis, KTH, School of Electrical Engineering and Computer Science (EECS).

[Yu et al. 2022] Yu, J., Moon, H., Chua, B.-L., and and, H. H. (2022). Hotel data privacy: strategies to reduce customers' emotional violations, privacy concerns, and switching intention. *Journal of Travel & Tourism Marketing*, 39(2):215–227.

[Zhang et al. 2021] Zhang, C., Xie, Y., Bai, H., Yu, B., Li, W., and Gao, Y. (2021). A survey on federated learning. *Knowledge-Based Systems*, 216:106775.

[Zhang et al. 2024] Zhang, Z., Zhu, H., and Xie, M. (2024). Differential privacy may have a potential optimization effect on some swarm intelligence algorithms besides privacy-preserving. *Information Sciences*, 654:119870.