UNIVERSIDADE FEDERAL DE MINAS GERAIS
Escola de Engenharia
Programa de Pós-Graduação em Engenharia Elétrica

Eliezer Timóteo da Silva Sanhá

ALGORITMO HÍBRIDO DE OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS PARA O APRENDIZADO FEDERADO DE REDES NEURAIS ARTIFICIAIS

Belo Horizonte

2024

Eliezer Timóteo da Silva Sanhá

ALGORITMO HÍBRIDO DE OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS PARA O APRENDIZADO FEDERADO DE REDES NEURAIS ARTIFICIAIS

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais, como requisito parcial à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. Frederico Gadelha Guimarães

Belo Horizonte
2024

Sanhá, Eliezer Timóteo da Silva.
S226a Algoritmo híbrido de otimização por enxame de partículas para o aprendizado federado de redes neurais artificiais [recurso eletrônico] / Eliezer Timóteo da Silva Sanhá. - 2024.
1 recurso online (73 f : il., color.) : pdf.

Orientador: Frederico Gadelha Guimarães.

Dissertação (mestrado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Bibliografia: f. 65-73.

1. Engenharia elétrica - Teses. 2. Aprendizado do computador - Teses. 3. Algoritmos - Teses. 4. Otimização - Teses. 5. Redes neurais (Computação) - Teses. 6. Redes neurais convolucionais - Teses. I. Guimarães, Frederico Gadelha. II. Universidade Federal de Minas Gerais. Escola de Engenharia. III. Título.

CDU: 621.3(043)

Ficha catalográfica elaborada pelo bibliotecário Reginaldo César Vital dos Santos CRB/6 2165
Biblioteca Prof. Mário Werneck, Escola de Engenharia da UFMG

UNIVERSIDADE FEDERAL DE MINAS GERAIS

ESCOLA DE ENGENHARIA

COLEGIADO DO CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

FOLHA DE APROVAÇÃO

"Algoritmo Híbrido de Otimização por Enxame de Partículas para o Aprendizado Federado de Redes Neurais Artificiais"

ELIEZER TIMOTEO DA SILVA SANHA

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Mestre em Engenharia Elétrica.

Aprovada em 10 de dezembro de 2024.

Por:

Prof. Dr. Frederico Gadelha Guimarães
DCC (UFMG) - Orientador

Prof. Dr. Jaime Arturo Ramírez
DEE (UFMG)

Prof. Dr. Marcelo Azevedo Costa
DEP (UFMG)

Prof. Dr. Demostenes Zegarra Rodriguez
DCC (UFLA)

Documento assinado eletronicamente por **Frederico Gadelha Guimaraes**, **Professor do Magistério Superior**, em 10/12/2024, às 12:43, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Marcelo Azevedo Costa**, **Professor do**

Magistério Superior, em 18/12/2024, às 16:21, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Demóstenes Zegarra Rodríguez**, **Usuário Externo**, em 19/12/2024, às 10:46, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Jaime Arturo Ramirez**, **Membro**, em 29/12/2024, às 18:03, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3795871** e o código CRC **50D6F6A9**.

Referência: Processo nº 23072.268801/2024-16 SEI nº 3795871

Aos meus familiares e amigos pelo apoio e por sempre acreditarem em mim.

AGRADECIMENTOS

Agradeço a Deus pela saúde e força proporcionada, durante momentos difíceis, para continuar lutando para atingir os meus objetivos. Agradeço aos meus familiares pelo amor, carinho e apoio demonstrado durante estes anos, que mesmo eu estando longe de casa, me dá força e vontade para sempre seguir em frente.

Agradeço aos meus amigos pelo apoio e incentivo durante todo esse período de mestrado. Agradeço ao Fabricio Javier Erazo-Costa pela ajuda durante o processo da escrita do trabalho. Gostaria de agradecer o apoio de todos os colegas do MINDS e FutureLab.

Agradeço especialmente ao meu orientador Prof. Dr. Frederico Gadelha Guimarães, pela orientação durante todo o período de mestrado, pela dedicação, pela paciência, ensino e incentivo durante todo este processo.

Agradeço a toda equipe do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais pela oportunidade de desenvolver o trabalho, também agradeço a FAPEMIG e a FUNDEP pelo apoio financeiro, via projeto de pesquisa e inovação *Programa de Aprendizado de Máquina Federado para Veículos Conectados*.

Este trabalho só foi possível devido à ajuda de todos vocês.

“O aprendizado nunca é finito, seja para humanos ou máquinas.”
(Inspirado em John McCarthy)

RESUMO

Os algoritmos de gerenciamento do treinamento de Aprendizado Federado (FL) comumente desconsideram o equilíbrio entre desempenho e custo de comunicação. Portanto, neste estudo propomos um novo método de treinamento federado denominado FLPSO-SGD que é baseado no algoritmo híbrido de otimização por enxame de partículas-Gradiente Descendente Estocástico (PSO-SGD) visando considerar esses aspectos. Em contraste com as técnicas clássicas de treinamento de FL, o método proposto neste trabalho recebe os erros dos clientes em vez dos parâmetros dos modelos, enquanto o algoritmo PSO-SGD realiza o treinamento no cliente. Os algoritmos foram avaliados em problemas de classificação utilizando as bases de dados da UC Irvine para o PSO-SGD, e a base de dados CIFAR-10, também foi utilizada com PSO-SGD e FLPSO-SGD em diferentes arquiteturas de redes neurais convolucionais. Os resultados destacam o desempenho promissor do algoritmo PSO-SGD. Além disso, a metodologia federada FLPSO-SGD demonstrou melhores desempenhos nos treinamentos de modelos globais em comparação com as técnicas FedAvg e FedPSO. Esses resultados se destacam, sobretudo, nas aplicações com os dados cuja amostras não são independentes e identicamente distribuídos (dados não IID). Os resultados sugerem que FLPSO-SGD é uma alternativa eficaz para treinamentos em FL, especialmente em aplicações com redes de comunicação com largura de banda restrita nos clientes.

Palavras-chave: aprendizado de máquina; redes neurais artificiais; redes neurais convolucionais; aprendizado federado; otimização por enxame de partículas.

ABSTRACT

The algorithms for managing Federated Learning (FL) training commonly overlook the balance between performance and communication cost. Therefore, in this study, we propose a new federated training method called FLPSO-SGD, based on the hybrid Particle Swarm Optimization-Stochastic Gradient Descent (PSO-SGD) algorithm, to address these aspects. In contrast to classical FL training techniques, the proposed method receives client errors instead of model parameters, while the PSO-SGD algorithm performs training on the client side. The algorithms were evaluated on classification problems using UC Irvine datasets for PSO-SGD, and the CIFAR-10 dataset was also used with both PSO-SGD and FLPSO-SGD across different convolutional neural network architectures. The results highlight the promising performance of the PSO-SGD algorithm. Moreover, the federated methodology FLPSO-SGD demonstrated better performance in training global models compared to the FedAvg and FedPSO techniques. These results are particularly notable in applications involving data that are not independent and identically distributed (non-IID data). The findings suggest that FLPSO-SGD is an effective alternative for FL training, especially in applications with communication networks that have restricted bandwidth on the client side.

Keywords: machine Learning; Neural Networks; convolutional neural networks; federated learning; particle swarm optimization.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AdaBest	*Adaptive Bias Estimation*
AdaGrad	*Adaptive Gradient Algorithm*
Adam	*Adaptive Moment Estimation*
AE	Algoritmos Evolucionários
ANN	*Artificial Neural Network*
AVC	Acidente Vascular Cerebra
BERT	*Bidirectional Encoder Representations from Transformers*
BP	*Backpropagation*
BPNN	*Back-Propagation Neural Network*
CNN	*Convolutional Neural Network*
DDPG	*Deep Deterministic Policy Gradient*
DE	*Differential Evolution*
DEPSO	*Differential Evolution with Particle Swarm Optimization*
DL	*Deep Learning*
DNN	*Deep Neural Network*
EC	*Evolutionary Computation*
FedAvg	*Federated Averaging*
FedNova	*Federated Normalized Averaging*
FedProx	*Federated Proximal*
FedPSO	*Federated Particle Swarm Optimization*
FNN	*Feedforward Neural Network*
FFNN	*Feed Forward Neural Network*
FEM	*Finite Element Method*

FL	*Federated Learning*
FL-PSO	*ederated Learning with PSO*
FLPSO-SGD	*Federated Learning PSO with SGD*
FTL	*Federated Transfer Learning*
GA	*Genetic algorithm*
GAN	*Generative Adversarial Networks*
gBest PSO	*Global Best PSO*
GD	*Gradient Descent*
GPU	*Graphics Processing Unit*
HGAPSO	*Hibrid of GA and PSO*
HMS	*Human Mental Search*
IA	Inteligência Artificial
IC	Inteligência Computacional
IID	Independentes e Identicamente Distribuídas
ILSVRC	*ImageNet Large Scale Visual Recognition Challenge*
IoT	*Internet of Things*
lBest PSO	*Local Best PSO*
LSTM	*Long Short-Term Memory*
ML	*Machine Learning*
MPL	*Multilayer perceptron*
NLP	*Natural Processing Language*
NN	*Neural Network*
PSO	*Particle Swarm Optimization*
PSO-GA	*Particle Swarm Optimization with Genetic Algorithm*
RFNN	*Random Feedforward Neural Network*
RMSE	*Root Mean-Square Error*

RMSProp	*Root Mean Square Propagation*
RNA	Rede Neural Artificial
RNN	*Recurrent Neural Network*
SAC	*Soft Actor-Critic*
SCAFFOLD	*Stochastic Controlled Averaging for Federated Learning*
SLP	*Single Layer Perceptron*
SGD	*Stochastic Gradient Gescent*
SI	*Swarm Intelligence*

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização

Os avanços tecnológicos têm possibilitado que uma variedade de dispositivos registrem e manipulem dados em diversos contextos, desde ambientes industriais e veiculares, até setores financeiros, hospitalares, residências, incluindo diversos dispositivos de borda (DARWISH; HASSANIEN; DAS, 2020). Nesse cenário, a utilização de ferramentas de Aprendizado de Máquina (ML, do inglês *Machine Learning*) torna-se indispensável para a manipulação, processamento e extração de informações úteis a partir de uma grande quantidade de dados, muitas vezes complexos (ALPAYDIN, 2020).

Algumas utilizações de ferramentes de ML em diversos campos e em diferentes tarefas importantes podem ser: ***Visão Computacional e Reconhecimento de Imagens***, Redes Neurais convolucionais (CNNs, do inglês *Convolucional Neural Networks*) têm dominado a visão computacional, destacando-se em tarefas como classificação de imagens, detecção de objetos e segmentação semântica, (HE; LIN, 2016; TAN; LE, 2019; SZEGEDY et al., 2016). ***Processamento de Linguagem Natural (NLP)***, transformadores, como o modelo BERT (do inglês *Bidirectional Encoder Representations from Transformers*), têm revolucionado o processamento de linguagem natural, alcançando avanços notáveis em tarefas como tradução automática, sumarização de texto e compreensão de linguagem natural (DEVLIN et al., 2018). ***Aprendizado por Reforço***, aplicações em aprendizado por reforço têm se destacado, com avanços em algoritmos como o DDPG (do inglês *Deep Deterministic Policy Gradient*) e o SAC *(do inglês* Soft Actor-Critic*)*, utilizados em robótica, jogos e controle de sistemas complexos (GU et al., 2016; HAARNOJA et al., 2018). No âmbito da ***Saúde e Diagnóstico Médico,*** redes neurais têm sido utilizadas para o diagnóstico médico, como na detecção de câncer em imagens de radiologia e patologia. Modelos baseados em redes neurais têm mostrado promissoras precisões em diversas especialidades médicas (ESTEVA et al., 2017; BEJNORDI et al., 2017). ***Finanças e Previsões***, redes neurais (e.g. Redes Recorrentes) são amplamente utilizadas em finanças para previsões de séries temporais, detecção de fraudes e tomada de decisões de investimento. Modelos como LSTM (Long Short-Term Memory) e GRU (Gated Recurrent Unit) têm se destacado nessas aplicações (HOCHREITER; SCHMIDHUBER, 1997; CHUNG et al., 2014). ***Indústria Automotiva e Automação***, redes neurais desempenham um papel crucial em sistemas de condução autônoma, utilizando CNNs para detecção de objetos, reconhecimento de semáforos e planejamento de trajetória (BOJARSKI et al., 2016). ***Geração de Conteúdo e Arte***, redes generativas, como GANs *(Generative Adversarial Networks)*, têm sido aplicadas na geração de arte, música e até mesmo na

criação de rostos realistas. O *Deep Art*, por exemplo, utiliza redes neurais para transformar fotografias em obras de arte (GOODFELLOW et al., 2014). Em todos esses campos e tarefas são utilizadas ferramentas de ML (por exemplo, arquiteturas de redes neurais, redes neurais profundas e algoritmos de treinamento), para o treinamento de modelos que possam atingir os objetivos propostos por cada campo e tarefas realizadas (ALPAYDIN, 2020).

Segundo Moshawrab et al. (2023b), quanto à arquitetura do sistema de ML, esta pode ser centralizada ou descentralizada (ou Aprendizado Federado, FL do inglês *Federated Learning*). O treinamento do ML centralizado com vários dispositivos envolve a transmissão e coleta centralizada de dados em um ponto específico, onde um servidor utiliza estas informações para calcular um modelo global. Apesar da eficácia desta abordagem, ela apresenta desafios em termos de privacidade e segurança de dados e eficiência computacional. Para resolver estes problemas, o trabalho de Konečnỳ et al. (2016) apresentou o conceito de FL, uma nova configuração de ML que permite que o treinamento seja colaborativo e com dados distribuídos. Neste paradigma, os clientes utilizam exclusivamente dados existentes no próprio dispositivo para calcular modelos locais. Ao contrário do treinamento centralizado, apenas os parâmetros (ou gradientes) dos modelos locais são transmitidos ao servidor para gerenciar o treinamento global utilizando as informações enviadas por cada cliente. O processo de encontrar um modelo global é realizado por um algoritmo de treinamento em FL (técnicas de agregação ou algoritmos de agregação). Nesta abordagem, os clientes mantêm a privacidade dos dados, porque estes não são mais transmitidos e são processados localmente. Com isso, o FL apresenta importantes benefícios, em termos de segurança e privacidade dos dados, entretanto, ainda existem desafios a serem superados, como o custo de comunicação (transmissão de uma grande quantidade de parâmetros), a latência no treinamento, a sobrecarga da rede, a própria eficiência dos modelos treinados, segurança dos modelos e a defesa contra os ataques (HAMER; MOHRI; SURESH, 2020).

1.2 Motivação

Vários algoritmos de treinamento em FL têm sido propostos com o objetivo de encontrar um modelo global de forma eficiente a partir das informações enviadas pelos clientes nos treinamentos locais. O algoritmo de agregação da Média Federada (FedAvg, do inglês *Federated Averaging*) foi a primeira técnica a ser desenvolvida visando o cálculo do modelo global pela média ponderada dos parâmetros enviados pelos dispositivos (MCMAHAN et al., 2017). Outros métodos foram propostos a partir de FedAvg, como o FedProx, (do inglês *Federated Proximal*), que permite trabalhar com dados não independentes e identicamente distribuídos atenuando as influências das heterogeneidades estatísticas no modelo global através de uma regularização proximal (LI et al., 2020b).

Outro algoritmo proposto a partir do FedAvg, foi SCAFFOLD (do inglês **Stochastic Controlled Averaging for Federated Learning**), que aprimora os algoritmos anteriores acelerando a convergência e diminuindo o desvio do cliente (resultados globais ineficientes gerados devido ao erro ou inadequação do treinamento local de um ou mais clientes) (KARIMIREDDY et al., 2020). Estas técnicas, dependendo de números de clientes ativos na rodada de comunicação, exigem o envio de uma grande quantidade de parâmetros dos modelos para o servidor. E apesar dessas técnicas ainda serem amplamente utilizadas, e de terem bons resultados, a transmissão de uma grande quantidade de parâmetros gera altos custos de comunicação (HAMER; MOHRI; SURESH, 2020), sobretudo em redes com ambientes de comunicação restrita entre clientes e servidor (HAMER; MOHRI; SURESH, 2020).

Para tratar este problema, Park, Suh e Lee (2021) apresentaram FedPSO, uma nova abordagem de treinamento para FL. Este método emprega o algoritmo meta-heurístico Otimização por Enxame de Partículas (PSO, do inglês *Particle Swarm Optimization*), no qual os clientes enviam as métricas de acurácias ou erros para o servidor, em lugar dos parâmetros. O servidor identifica o cliente com melhor desempenho e solicita exclusivamente a este dispositivo seus parâmetros adquiridos no treinamento local usando Gradiente Descendente Estocástico (SGD, do inglês *Stochastic Gradient Descent*). Esses parâmetros são considerados como globais e, em seguida, transmitidos para atualizar os parâmetros de um grupo de clientes selecionados.

Apesar do método apresentado no trabalho de Park, Suh e Lee (2021) conseguir reduzir a quantidade de parâmetros transmitidos do cliente para o servidor em cada rodada de comunicação, o método apresenta um desempenho em termos de erro e acurácia similar ao FedAvg, ou seja FedPSO teve um desempenho mediano, devido ao treinamento local dos clientes ser o mesmo que o FedAvg. O treinamento local dos clientes em FedAvg é feita por SGD e suas variantes que são amplamente adotadas na otimização de parâmetros de Redes Neurais Artificiais (RNAs) devido à sua eficiência e escalabilidade (HUANG et al., 2019). Esta característica os torna a escolha predominante para abordagens de treinamentos locais dos clientes em FL. No entanto, o SGD pode encontrar dificuldades de convergência em mínimos locais, alta variabilidade, e consequentemente uma convergência lenta, especialmente em funções não diferenciáveis (YADAV et al., 2020). Em um contexto federado, isso pode levar ao surgimento do desvio de cliente e diminuir o desempenho dos modelos globais (MOSHAWRAB et al., 2023b). Apesar de FedPSO oferecer uma solução na redução no custo de comunicação para aplicações de FL, essa metodologia utiliza PSO exclusivamente para gerenciar o treinamento federado global, enquanto o treinamento local está baseado (da mesma forma que FedAvg) no SGD.

Para superar estas limitações, este trabalho propõe uma nova metodologia de FL denominada FLPSO-SGD que visa melhorar tanto o desempenho do algoritmo federado

quanto a redução do custo de comunicação. Para isso, a proposta do treinamento local dos clientes é a utilização de um novo algoritmo híbrido denominado PSO-SGD. Neste método, o SGD se torna uma componente adicional da equação de atualização da velocidade da partícula no PSO, isto é, a utilização da versão *Global Best PSO (gBest PSO)* atualiza o modelo global em direção do melhor cliente (partícula), enquanto o treinamento de FL utiliza o paradigma de envio das métricas do erro de cada cliente (PARK; SUH; LEE, 2021). A escolha de PSO e SGD se sustentam com a base na natureza de PSO de ser capaz de explorar rapidamente um espaço de busca e obter alto desempenho e sua capacidade de hibridização com outros algoritmos de otimização, o que pode melhorar consideravelmente o desempenho de PSO (YADAV et al., 2020). No âmbito de FL, o treinamento local com PSO consegue apresentar algumas vantagens adicionais, isto é, para dados não homogêneos ou ataques contra o modelo global, provenientes de "clientes maliciosos", devido ao seu fator de inércia e componente social o PSO consegue atenuar em certa medida as mudanças abruptas que possam ocorrer no modelo global devido à influência do treinamento local do cliente (DARWISH; HASSANIEN; DAS, 2020). Com isso a hibridização PSO-SGD consegue aproveitar as vantagens dos dois métodos sem suas desvantagens inerentes.

Os métodos propostos foram comparados com os métodos atuais da literatura. Inicialmente validou-se o PSO-SGD para o treinamento local dos clientes, através de treinamento, teste e comparações com diferentes metologias de treinamento, como Adagrad Dean et al. (2012), RMSProp Tieleman e Hinton (2012) e Adam Kingman e Ba (2015), em diferentes problemas de classificação de ML. Para o treinamento federado completo o FLPSO-SGD obteve resultados relativamente superiores ao FedPSO Park, Suh e Lee (2021) e FedAvg (MCMAHAN et al., 2017) em diferentes arquiteturas convolucionais. Os resultados demonstraram um desempenho promissor do algoritmo PSO-SGD e da metodologia federada FLPSO-SGD oferecendo uma nova perspectiva no treinamento federado, proporcionando melhorias significativas no desempenho e na redução do custo de comunicação.

1.3 Objetivos

O principal objetivo deste trabalho é pesquisar e propor o algoritmo de treinamento de aprendizado federado FLPSO-SGD, para o treinamento de redes neurais artificiais, especialmente em arquiteturas de redes neurais profundas. Para atingir o pretendido, foram estabelecidas algumas metas específicas.

- pesquisa e desenvolvimento de algoritmo híbrido de otimização PSO-SGD para a validação da técnica de treinamento local dos clientes.
- Implementação computacional de PSO-SGD em problemas clássicos de ML e em um

problema de visão computacional.

- Avaliar o desempenho do algoritmo desenvolvido, através da comparação com outros métodos de otimização pertencentes ao estado-da-arte.
- Estabelecer e desenvolver o modelo federado do algoritmo PSO-SGD, que seria a implementação do FLPSO-SGD em Rede Neurais Convolucionais (CNN, do inglês *Convolutional Neural Network*).
- Avaliação do desempenho do FLPSO-SGD em diferentes arquiteturas do tipo CNN, comparando-o ao desempenho de outros métodos de treinamento em aprendizado federado.
- Avaliação do desempenho do FLPSO-SGD em diferentes arquiteturas CNN com dados não IID.

1.4 Organização do Trabalho

O restante do presente trabalho está organizado da seguinte forma: O capítulo 2 descreve a fundamentação teórica e os trabalhos relacionados. O capítulo 3 apresenta os algoritmos propostos no presente trabalho, isto é, um algoritmo de aprendizado federado denominado de FLPSO-SGD, e PSO-SGD para o treinamento local dos clientes. O capítulo 4 descreve os experimentos realizados, resultados e discussões, incluindo os conjuntos de dados utilizadas e as configurações experimentais. O capítulo 5 conclui o trabalho, apresentando as limitações e perspectivas futuras.

2 FUNDAMENTAÇÃO E TRABALHOS RELACIONADOS

O presente capítulo debruça sobre os conceitos fundamentais que norteiam o presente trabalho. Iniciando com os fundamentos dos algoritmos de aprendizado de máquina, aplicados ao treinamento de redes neurais artificiais. Em seguida, aborda-se as principais técnicas de otimização baseadas em gradiente descendente estocástico. Da mesma forma, apresenta-se os conceitos fundamentais do algoritmo de Otimização por Enxame de Partículas, que nos últimos anos tem se mostrado eficaz no treinamento de redes neurais artificias (DARWISH; HASSANIEN; DAS, 2020; OJHA; ABRAHAM; SNÁŠEL, 2017). O capítulo ainda traz os conceitos e as abordagens relacionadas ao aprendizado federado (KONEČNỲ et al., 2016). Por fim, são apresentados alguns trabalhos relacionados aos métodos propostos no capítulo seguinte.

2.1 Algoritmos de Otimização Baseados em Gradiente Descendente

Os algoritmos de otimização são essenciais na etapa de treinamento dos modelos de ML. Essas técnicas utilizam os dados de treinamento para minimizar uma função de custo, ajustando os parâmetros do modelo (pesos e vieses), tendo como objetivo a obtenção de um modelo robusto, com um bom desempenho e capacidade de generalização a entrada de novos dados (MEHMOOD; AHMAD; WHANGBO, 2023). A escolha de um algoritmo de otimização adequado depende do tipo de problema, da eficiência na redução dos erros e do tempo de treinamento dos modelos (CHOI et al., 2019).

Os algoritmos de otimização baseados em Gradiente Descendente (GD) são métodos que têm sido utilizados amplamente nas redes neurais artificiais (NEWTON; YOUSEFIAN; PASUPATHY, 2018). Isto se deve à facilidade de utilização e sua alta efetividade no treinamento dos modelos de ML (MEHMOOD; AHMAD; WHANGBO, 2023). O GD funciona da seguinte forma: inicializa os parâmetros do modelo; realiza o cálculo do gradiente da função de custo em relação aos parâmetros; e atualiza a Equação (2.1); repete o processo até cumprir com as condições de parada.

$$\theta_{i+1} = \theta_i - \eta \nabla L(\theta_i), \tag{2.1}$$

onde θ_i são os parâmetros do modelo na i-ésima iteração, η é a taxa de aprendizado (o tamanho do passo em direção ao mínimo da função) e $\nabla L(\theta_i)$ é o gradiente da função de custo em relação aos parâmetros do modelo.

O SGD (2.2) é baseado no GD (ROBBINS; MONRO, 1951). A diferença entre os dois consiste na forma como o cálculo do gradiente é efetuado. Isto é, o GD usa todo o

conjunto de amostras de dados do treinamento para o cálculo do gradiente da função de custo em cada iteração, o que pode elevar o custo computacional em relação ao tempo de treinamento. Enquanto que o SGD utiliza uma única amostra ou mini-lotes D_j por vez para fazer o cálculo do gradiente da função de custo em relação aos parâmetros. Esta modificação não só acelera o treinamento dos modelos, como também melhora a sua eficiência (NEWTON; YOUSEFIAN; PASUPATHY, 2018).

$$\theta_{i+1} = \theta_i - \eta \nabla L(\theta_i; D_j), \tag{2.2}$$

onde D_j representa um único par de amostra j ou mini-lote j dos dados. Segue abaixo uma lista com algumas técnicas baseadas em SGD, que são importantes para o presente trabalho.

- **SGD + Momentum**, Equação (2.4), combina o SGD com um termo de momentum, ou seja adiciona uma componente de velocidade v e um fator de momentum β, Equação (2.3). Este algoritmo é um aprimoramento do SGD com objetivo de acelerar o processo de convergência e consequentemente o treinamento dos modelos (POLYAK, 1964).

$$v_{i+1} = \beta \cdot v_i + (1 - \beta) \cdot \nabla L(\theta_i) \tag{2.3}$$

$$\theta_{i+1} = \theta_i - \eta \cdot v_{i+1} \tag{2.4}$$

  θ_i são os parâmetros do modelo na i-ésima iteração, $\nabla L(\theta_i)$ é o gradiente da função de custo L em relação aos parâmetros θ, η é a taxa de aprendizado, β é o coeficiente de momentum (geralmente próximo a 1) e v_i (velocidade) é o vetor de momentum na i-ésima iteração.

- **AdaGrad** (do inglês *Adaptive Gradient Algorithm)* dada pela Equação (2.5), surge como a solução para taxa de aprendizado η fixa, que é usada pelo SGD durante todo o processo de treinamento (DUCHI; HAZAN; SINGER, 2011). Ou seja, em alguns problemas ou em algumas situações, pode-se exigir atualizações de parâmetros mais lentas ou mais rápidas, nessas situações SGD pode oscilar ou retardar a convergência. O AdaGrad introduziu o conceito de taxa de aprendizado adaptativa, para suprir essas limitações ao ajustar automaticamente a taxa de aprendizado η para cada parâmetro baseado no histórico de gradientes acumulados (DEAN et al., 2012).

$$\theta_{i+1,k} = \theta_{i,k} - \frac{\eta}{\sqrt{G_{i,k} + \epsilon}} \cdot \nabla L(\theta_i)_k \tag{2.5}$$

  onde $G_{i,k}$ é a soma dos quadrados dos gradientes até a iteração i para o k-ésimo parâmetro, e ϵ é um termo de suavização para evitar divisão por zero.

- **RMSProp** (do inglês *Root Mean Square Propagation*) definida pela Equação (2.6), modifica o AdaGrad para dar menos peso a gradientes acumulados, evitando assim, uma taxa de aprendizado excessivamente reduzida. Nesse sentido, ao fazer uma média móvel dos quadrados de gradientes acumulados, evita que a taxa de aprendizado diminua excessivamente em situações de convergência de longo prazo em redes mais profundas (HINTON; SRIVASTAVA; SWERSKY, 2012; TIELEMAN; HINTON, 2012).

$$\theta_{i+1,k} = \theta_{i,k} - \frac{\eta}{\sqrt{\frac{1}{i}\sum_{\tau=1}^{i}(\nabla L(\theta_\tau)_i)^2 + \epsilon}} \cdot \nabla L(\theta_i)_k, \tag{2.6}$$

  onde $\sqrt{\frac{1}{i}\sum_{\tau=1}^{i}(\nabla L(\theta_\tau)_i)^2 + \epsilon}$ adapta η ao gradientes acumulados atuais.

- **Adam** (do inglês *Adaptive Moment Estimation*) representada pela Equação (2.9), é o algoritmo que combina características do Momentum (Equação (2.7) média móvel dos gradientes) e do RMSProp (Equação (2.8) média móvel dos quadrados dos gradientes) (KINGMA; BA, 2014). A incorporação do Momentum (2.7), facilita a suavização das oscilações em superfícies curvas, também estabiliza e acelera a convergência, enquanto que a combinação com RMSProp ajusta automaticamente a taxa de aprendizado de cada parâmetro (KINGMAN; BA, 2015).

$$m_{i+1} = \beta_1 m_i + (1 - \beta_1)\nabla L(\theta_i) \tag{2.7}$$

$$v_{i+1} = \beta_2 v_i + (1 - \beta_2)(\nabla L(\theta_i))^2 \tag{2.8}$$

$$\theta_{i+1} = \theta_i - \frac{\eta}{\sqrt{v_{i+1}} + \epsilon} \cdot m_{i+1} \tag{2.9}$$

  em que m_i é a média móvel dos gradientes, v_i é a média móvel dos quadrados dos gradientes, β_1 e β_2 são fatores de decaimento para m e v, respectivamente.

2.2 Otimização por Enxame de Partículas

Inteligência do Enxame (SI do inglês *Swarm Intelligence*) refere-se às abordagens meta-heurísticas que se inspiram nos comportamentos coletivos, considerados inteligentes, de grupos naturais ou artificiais de entidades que podem ser individualmente incapazes, mas que conseguem formar um sistema auto-organizado, produzindo assim um comportamento global inteligente. A Figura 1 ilustra uma revoada de pássaros em formações coordenadas, exemplificando um comportamento coletivo que fundamenta os conceitos de SI (CORNE; REYNOLDS; BONABEAU, 2012; SENGUPTA; BASAK; PETERS, 2018).

Figura 1 – Bando de pássaros voando em formações coordenadas, que sugerem um comportamento inteligente coletivo (imagem adquirida na Wikipédia).

A computação natural se desenvolveu consideravelmente nas últimas décadas com o surgimento do processamento paralelo, o que facilitou o aparecimento e consequentemente a implantação de algoritmos complexos capazes de fazer buscas eficientes em hiperespaços (SENGUPTA; BASAK; PETERS, 2018). Diversos algoritmos de computação natural têm sido propostos como o PSO (KENNEDY; EBERHART, 1995) que pertence ao grupo de SI; e as técnicas baseadas em Algoritmo genético (GA, do inglês *Genetic Algorithms*) (FORREST, 1996) e Evolução Diferencial (DE, do inglês *Differential Evolution*) (STORN; PRICE, 1997), que fazem parte dos algoritmos evolucionários. Estas abordagens permitem a resolução de problemas de funções complexas que são difíceis ou impossíveis de serem resolvidas por abordagens determinísticas como é caso de algoritmos baseados em GD (CORNE; REYNOLDS; BONABEAU, 2012).

Como na revoada apresentada na Figura 1, no PSO canônico existem dois componentes de extrema importância: componente cognitivo da partícula (tem a ver com a posição individual de um pássaro na revoada) e o componente social das partículas (tem a ver com a influência do comportamento da revoada sobre um pássaro). O componente cognitivo tem a ver com a posição individual de uma partícula (posição individual de um pássaro na revoada). Enquanto que o componente social tem a ver com a posição do enxame (toda a revoada na versão gBest, do inglês *Global Best PSO*) ou posição da vizinhança (um grupo de pássaros próximos na versão lBest, do inglês *Local Best PSO*) (KENNEDY; EBERHART, 1995; SHI; EBERHART, 1998; KENNEDY; MENDES, 2002).

No algoritmo meta-heurístico PSO (KENNEDY; EBERHART, 1995), cada partícula

representa uma solução candidata que é iterativamente ajustada no espaço de busca em direção aos valores ótimos. Esses ajustes são influenciados pelo comportamento cognitivo que corresponde à melhor posição encontrada pela própria partícula, e pelo comportamento social, que é determinado pela melhor posição encontrada por qualquer partícula do enxame ou em sua vizinhança. A atualização da velocidade da partícula é regida pela Equação (2.10), enquanto a atualização de sua posição no espaço de busca é regida pela Equação (2.11).

$$v_{k(i+1)} = wv_{ki} + c_1 r_{1ki}(pbest_{ki} - \theta_{ki}) + c_2 r_{2ki}(gbest_i - \theta_{ki}) \tag{2.10}$$

$$\theta_{k(i+1)} = \theta_{ki} + v_{k(i+1)}, \tag{2.11}$$

onde w é o fator da inércia, v representa a velocidade da partícula, r_1 e r_2 são números aleatórios entre [0, 1], c_1 e c_2 são coeficientes de aceleração cognitivo e social, respectivamente. A variável θ é a posição da partícula, *pbest* é a melhor posição encontrada pela partícula, *gbest* é a melhor posição encontrada entre as partículas, o índice k representa k-ésima partícula e o índice i representa i-ésima iteração. O PSO representada pela equação (2.10), corresponde a versão canônica *Global Best PSO*. O pseudo-código do Algoritmo 1, apresenta o funcionamento em detalhe do PSO canônico.

Algorithm 1 PSO Canônico

Inicializar cada partícula k com uma posição aleatória θ_k e uma velocidade inicial $\mathbf{v}_k$
Inicializar a melhor posição local $\mathbf{pbest}_k$ de cada partícula como sua posição inicial
Inicializar a melhor posição global **gbest** como a melhor posição entre todas as partículas
Enquanto i for menor que a quantidade de iterações estabelecidas
    Para cada partícula k
        Atualizar a velocidade: $\mathbf{v}_{ki} \leftarrow w \cdot \mathbf{v}_{ki} + c_1 \cdot r_1 \cdot (\mathbf{pbest}_{ki} - \theta_{ki}) + c_2 \cdot r_2 \cdot (\mathbf{gbest}_i - \theta_{ki})$
        Atualizar a posição: $\theta_i \leftarrow \theta_{ki} + \mathbf{v}_{ki}$
        Se a posição atual θ_{ki} é melhor que $\mathbf{pbest}_{ki}$
            Atualizar a melhor posição local: $\mathbf{pbest}_{ki} \leftarrow \theta_{ki}$
        Fim Se
        Se a posição atual θ_{ki} é melhor que $\mathbf{gbest}_i$
            Atualizar a melhor posição global: $\mathbf{gbest}_i \leftarrow \theta_{ki}$
        Fim Se
    Fim Para
Fim Enquanto
Retornar $\mathbf{gbest}_i$ como a melhor solução encontrada

A inicialização das partículas é importante para alcançar mais rápido os valores ótimos da função de custo. Por exemplo, se a velocidade inicial é zero o *pbest* e *gbest* desempenham um papel importante na busca inteligente na vizinhança da posição inicial, mas não favorecem a exploração de novas regiões no espaço de busca. Já a velocidade

do enxame inicializada contribui para a exploração mais ampla do espaço de busca (SHI; EBERHART, 1998). No entanto, é fundamental impor limites adequados à velocidade para garantir que o enxame não divirja. A seleção apropriada da velocidade máxima v_{max} é importante para manter o controle da exploração de espaço de busca: um v_{max} grande permite a exploração global do espaço de busca, enquanto um valor pequeno implica uma busca local e intensiva (KENNEDY; MENDES, 2002).

A componente de inércia do PSO wv_i, é composto por um coeficiente da inércia (ou peso da inércia) w que regulariza uma velocidade inicial (ou atual) da partícula v_i. O peso da inércia pode ser constante ou decaindo, Equação (2.12) apresenta a abordagem com peso da inércia w que decai linearmente, onde w_{max} e w_{min}, são os limites máximo e mínimo do w, respectivamente, i_{max} é a quantidade máxima de iteração e i é a iteração atual (SUGANTHAN, 1999; RATNAWEERA, 2002).

$$w_{decai} = (w_{max} - w_{min})\frac{i_{max} - i}{i_{max}} + w_{max} \tag{2.12}$$

Outra modificação do algoritmo canônico do PSO é estabelecer um coeficiente de constrição χ como uma forma de melhorar o controle da velocidade da partícula. A Equação (2.13) mostra o cálculo da expressão χ, enquanto que a Equação (2.14) mostra como χ modifica a Equação (2.10) que faz atualização de velocidade (CLERC; KENNEDY, 2002; CLERC, 1999).

$$\chi = \frac{2T}{|2 - \phi - \sqrt{(\phi(\phi - 4))}|} \tag{2.13}$$

$$v_{ki} = \chi[v_{ki} + \phi_1(pbest_{ki} - \theta_{ki}) + \phi_2(gbest_i - \theta_{ki})] \tag{2.14}$$

onde $\phi = \phi_1 + \phi_2$, $\phi_1 = c_1 + r_{1i}$, $\phi_2 = c_2 + r_{2i}$ e $\phi \geq 4$. O termo $T \in [0,\ 1]$ controla a exploração e aproveitamento do enxame.

A escolha dos coeficientes de aceleração, c_1 e c_2 são importantes para um funcionamento mais dinâmico e rápido do algoritmo. Porque seus valores determinam o quanto uma partícula deve ponderar ao se mover em direção ao seu fator cognitivo (*pbest*) ou ao fator social (*gbest*). A comunicação entre as partículas implica cooperação inerente, sugerindo que uma escolha imparcial dos coeficientes de aceleração os tornaria iguais, ou seja, todas as partículas no enxame possuem os mesmos c_1 e c_2. Em implementações específicas, pode-se definir $c_1 = 0$ ou $c_2 = 0$, fazendo com que as partículas dependam exclusivamente de seu próprio conhecimento ou apenas do conhecimento da melhor partícula no enxame. Desse modo, em funções e problemas multimodais, contendo várias regiões com mínimos locais, as partículas se beneficiarão de um equilíbrio entre os componentes sociais e cognitivos da aceleração (SENGUPTA; BASAK; PETERS, 2018). Normalmente, os valores de c_1 e c_2

podem ser constantes, no entanto, existem abordagens em que esses coeficientes variam, com crescimentos ou decaimentos lineares, conforme sugerido por Ratnaweera (2002). Estudos demonstraram que adaptações empíricas nos valores dos coeficientes de aceleração podem levar a melhorias no desempenho de diferentes grupos de enxames, um par ótimo para cada c_1 e c_2 pode ser determinado empiricamente para cada caso. Contudo, desvios significativos ou inicializações incorretas podem resultar em comportamentos divergentes (ZHANG; LIU; CLERC, 2003). Nos trabalhos Mendes, Kennedy e Neves (2003) e LIU et al. (2016), os autores estudaram as implicações de diferentes coeficientes sociais e cognitivos em um tamanho populacional fixo, quantificando o efeito da inclusão das experiências passadas de um indivíduo, tanto ao implementar o algoritmo com a partícula de interesse quanto sem ela.

Existem topologias ou configurações de PSO que se baseiam nas comunicações entre as partículas do enxame, isto é, um subconjunto de partículas com as quais uma partícula pode iniciar a comunicação (KENNEDY; MENDES, 2002). Como visto anteriormente, o PSO original delineou duas configurações que levaram a duas variantes principais do algoritmo: *gBest PSO*, que foi descrito na Equação (2.10), também é a versão utilizada no presente trabalho, e *lBest PSO* (KENNEDY, 1999).

A variante *lBest PSO*, agrupa um número específico do total de partículas na vizinhança de uma determinada partícula. Nessa configuração, cada vizinhança no enxame possui uma *lbest*, e consequentemente a equação de atualização de velocidade do PSO tem múltiplos atratores sociais, como mostra a Equação (2.15). Nessas condições, o enxame não é atraído para uma única melhor solução global, mas sim para uma combinação das melhores soluções locais (vizinhança) (KENNEDY, 1999; KENNEDY; MENDES, 2002). Desta forma, existe uma redução da velocidade de convergência, mas com um aumento significativo na probabilidade de encontrar ótimos globais. Diferentemente da variante *gBest PSO*, onde todas as partículas são influenciados pelo *gbest* do enxame na atualização de suas velocidades. Assim, levando a uma velocidade de convergência aumentada e a uma potencial estagnação em ótimos locais, isto é, se o verdadeiro ótimo global não estiver onde a melhor partícula do enxame está. Entretanto, na versão *lBest PSO* as partículas são influenciadas exclusivamente pelas melhores partículas *lbest* de suas respectivas vizinhanças Equação (2.15).

$$v_{k(i+1)} = wv_{ki} + c_1 r_{1ki}(pbest_{ki} - \theta_{ki}) + c_2 r_{2ki}(lbest_{ji} - \theta_{ki}), \tag{2.15}$$

onde $lbest_{ji}$ representa a melhor posição encontrada na j-ésima vizinhança em i-ésima iteração.

Anteriormente vimos que o componente social do PSO pode ser configurada como g*Best PSO*, que também chamada de configuração Estrela, ver Figura 2(a) e Equação (2.10) (KENNEDY; EBERHART, 1995; PARSOPOULOS; VRAHATIS, 2005). Também se

abordou a configuração *lBest PSO*, também conhecida de configuração de Anel, ver Figura 2(b), Equação (2.15) (KENNEDY, 1999). Entretanto, existem outras configurações comuns da vizinhança no PSO. Por exemplo, a configuração *Wheel* (Roda), Figura 2(c), onde o enxame possui uma partícula focal que exerce a influência sobre as demais partículas nas periferias (KENNEDY; MENDES, 2002; MENDES; KENNEDY; NEVES, 2003). A configuração de Von Neumann, Figura 2(d), onde as partículas têm 4 ou mais vizinhos imediatos, e.g.: acima, abaixo, à esquerda e à direita (KENNEDY; MENDES, 2002). A Figura 2(e) mostra uma configuração social de 4 agrupamentos de partículas (*4-Clusters*) (MENDES; KENNEDY; NEVES, 2003; XU; PI, 2020), nesta configuração cada partícula de um determinado agrupamento interage exclusivamente com as partículas do mesmo grupo. O que facilita a exploração paralela de várias regiões no espaço de busca.

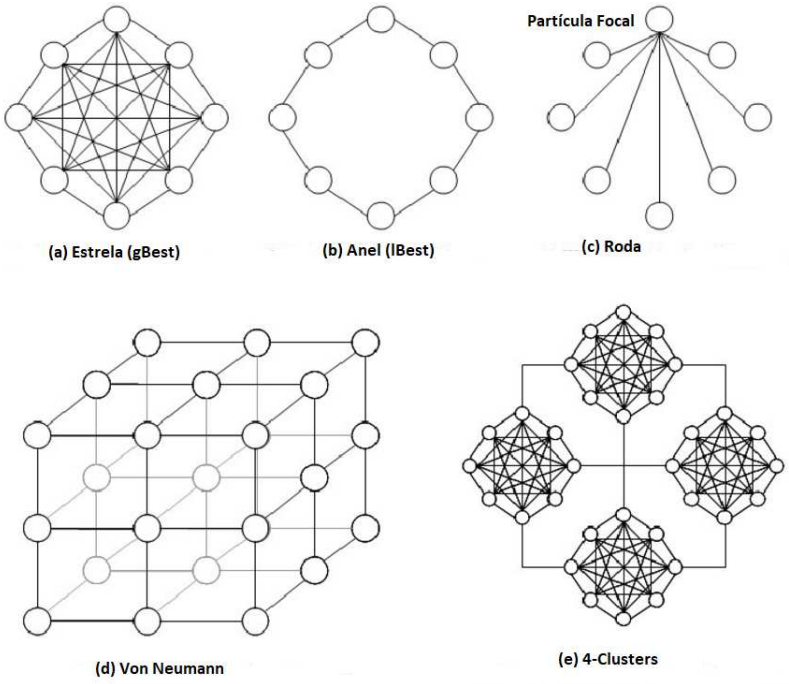

Figura 2 – Configurações de estruturas de componente social aplicadas ao PSO (ZAVALA; AGUIRRE; DIHARCE, 2007).

2.2.1 PSO Híbrido

A hibridização de PSO tem sido uma prática mais frequente a partir dos anos 2000, e com um crescimento significativo nas últimas duas décadas. A integração de algoritmos de otimização meta-heurísticos ou determinísticos tem se mostrado eficaz na resolução de vários problemas de otimização ou/e da inteligência computacional (SHAMI et al., 2022).

A combinação do PSO com o Algoritmo Genético (GA do inglês *Genetic Algorithm*) é uma estratégia poderosa para melhorar o desempenho de ambos os métodos quando utilizados de forma separada. Existem diversas abordagens de hibridização entre o PSO

e o GA, que incluem a aplicação sequencial dos dois algoritmos, também sua aplicação em paralelo. Ou ainda, a incorporação de operadores do GA, como seleção, mutação e recombinação, dentro de uma estrutura do PSO. Robinson, Sinton e Rahmat-Samii (2002) utilizaram os dois algoritmos em alternância para variar o critério de parada. Eles aplicaram essa alternância para permitir que os algoritmos se auto ajustassem, melhorando seu desempenho quando um deles falhava ou encontrava dificuldades durante as iterações. Além disso, no trabalho de Shi et al. (2005), o algoritmo PSO tem como critério de parada um número específico de iterações e as melhores partículas do PSO passam a compor a população inicial do GA. Enquanto que as posições vazias são preenchidas usando gerações aleatórias. Assim, a diversidade da população é preservada e o desempenho é semelhante ao final da primeira fase. Pesquisas complementares de Yang, Chen e Zhao (2007) propuseram o algoritmo evolutivo híbrido (HEA) que é baseado em PSO-GA. Esta estratégia de evolução de partículas utiliza um mecanismo de duas fases, onde o processo de evolução é acelerado pelo uso de PSO e a diversidade é mantida pelo uso de GA. Posteriormente, os autores Li, Xu e Shi (2008) aplicaram esse método para otimizar três problemas não restritos e três problemas restritos, utilizando mecanismos de seleção como um classificador não linear para gerar descendentes de pais híbridos de GA-PSO. Cada estágio do processo foi executado separadamente, utilizando os algoritmos GA e PSO. Ainda há outros estudos que propuseram uma abordagem nebulosa Valdez, Melin e Castillo (2009) para a hibridização PSO-GA, onde o algoritmo nebuloso determina se as partículas são gerenciadas pelo GA ou PSO, como também ajusta os seus parâmetros e toma de decisões relacionadas. Também, (GHAMISI; BENEDIKTSSON, 2014) introduziram uma metodologia de seleção de características ao hibridizar o GA e o PSO. O método foi testado e obteve bons desempenhos no conjunto de dados *hiperespectral Indian Pines*, bem como em tarefas de detecção de estradas.

Existem várias aplicações do algoritmo híbrido GA-PSO. Por exemplo: a quantificação de métricas em áreas de produção alimentares (BENVIDI et al., 2017); A estimação da demanda de energia elétrica (YU; WEI; WANG, 2012; LI et al., 2018); a classificação de módulos de software propensos a falhas, utilizando programação orientada a objetos Moussa e Azar (2017); a otimização de unidades de inspeção de pavimentos asfálticos em redes extensas (NIK; NEJAD; ZAKERI, 2016); e uma versão discreta de PSO com operadores de GA incorporados para fins de agrupamento, no qual o operador de GA inicia a reprodução quando as partículas estagnam (PREMALATHA; NATARAJAN, 2009). Além disso, vale a pena destacar que o método de hibridização de PSO-GA, tem sido empregado com Redes Neurais Artificiais RNAs (ANNs/ NNs do inglês *Artificial Neural Networks/Neural Netwoks*), Máquina de Vetores de Suporte (SVM), Lógica Fuzzy, entre outras (VALDEZ; MELIN; CASTILLO, 2009; GHAMISI; BENEDIKTSSON, 2014; LI et al., 2018; KRINK; LØVBJERG, 2002; CONRADIE; MIIKKULAINEN; ALDRICH, 2002).

A hibridização de PSO e Evolução Diferencial (DE do inglês *Differential Evolution*)

também é uma opção, que tem sido amplamente utilizada. A DE (STORN, 1995) é uma meta-heurística popular e eficaz para resolver problemas de otimização global. Existem várias abordagens de hibridização de DE com PSO na literatura como: a utilização de DE intermitentemente para mover partículas de áreas de desempenho inferior para aquelas de melhor desempenho Hendtlass (2001). Também, Zhang e Xie (2003) introduziram outra variante de um híbrido DEPSO em que a utilização de cada algoritmo realizada de forma intercalada em lugar de executar ambos algoritmos ao mesmo tempo. Isto é, o papel de DE nessa hibridização é de fazer uma intervenção periódica para gerar diversidade na população, possibilitando assim o escape de mínimos locais. Posteriormente, Talbi e Batouche (2004) empregaram um DEPSO para abordar o problema de registro de imagem multimodal de corpo rígido, ou seja, usar DEPSO para encontrar a melhor transformação rígida de duas imagens multimodais com a máxima de informações mútuas. Adicionalmente, Hao, Guo e Huang (2007) utilizaram atualizações seletivas para as posições das partículas, combinando parcialmente a abordagem DE com a abordagem PSO, para resolver vários problemas benchmark de otimização. Por outro lado, Das, Abraham e Konar (2008) eliminaram o coeficiente cognitivo da equação de atualização de velocidade em PSO, substituindo-o por um vetor de diferença ponderada entre as posições de duas partículas escolhidas aleatoriamente da população com DE. Ademais, Luitel e Venayagamoorthy (2008) utilizaram um otimizador DEPSO para projetar filtros de resposta ao impulso finito FIR. Enquanto que Vaisakh, Sridhar e Murthy (2009) propuseram um algoritmo DEPSO para alcançar o despacho ótimo de potência reativa, com redução de potência e aprimoramento da estabilidade da tensão. Por sua vez, Huang et al. (2009) estudaram a análise inversa de parâmetros mecânicos utilizando o DEPSO-ParallelFEM, um método híbrido que busca combinar as vantagens do DE, PSO e do Método dos Elementos Finitos (FEM).

2.3 Aprendizado Federado

Segundo Moshawrab et al. (2023a), quanto à taxonomia de arquiteturas, os algoritmos de ML são categorizados em Centralizado e Federado como ilustra a Figura 3. Esta seção apresenta o FL e a sua diferença com aprendizado centralizado.

No aprendizado centralizado, os clientes enviam seus dados para um único servidor, onde as informações são agregadas e armazenadas. O servidor então utiliza esse conjunto centralizado de dados para treinar o modelo. A Figura 4 ilustra como cada cliente (dispositivo de borda) envia suas informações, que são centralizadas no servidor para o treinamento do modelo. Embora, esse tipo de treinamento permita o uso de um conjunto amplo e diversificado de dados, ele enfrenta limitações, como a falta de privacidade no compartilhamento dos dados e altos custos de comunicação relacionados à transmissão dos dados (MOSHAWRAB et al., 2023a).

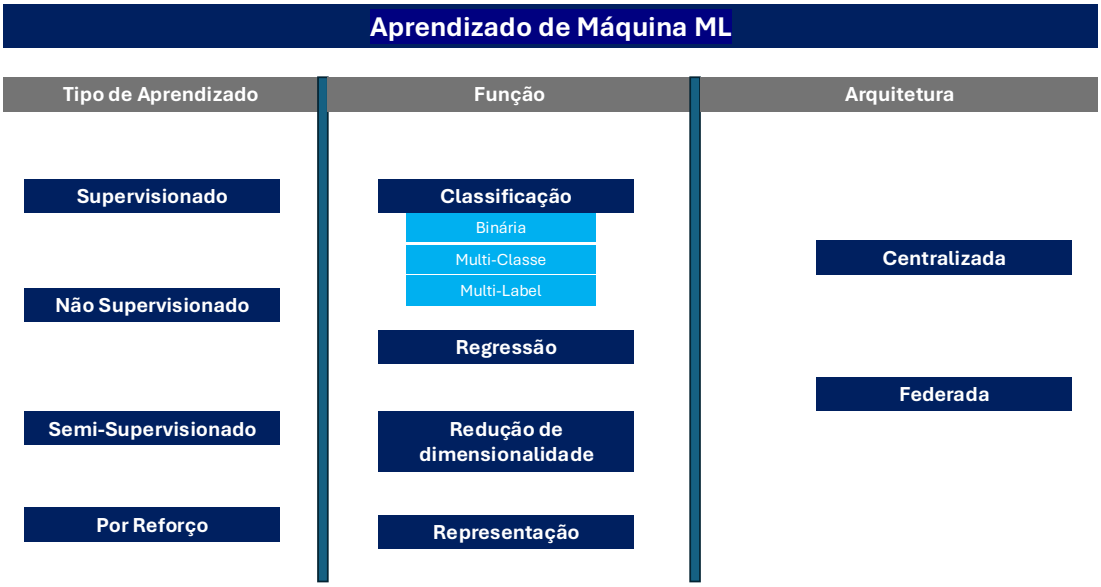

Figura 3 – Representação taxonômica simplificada dos algoritmos de ML (MOSHAWRAB et al., 2023a).

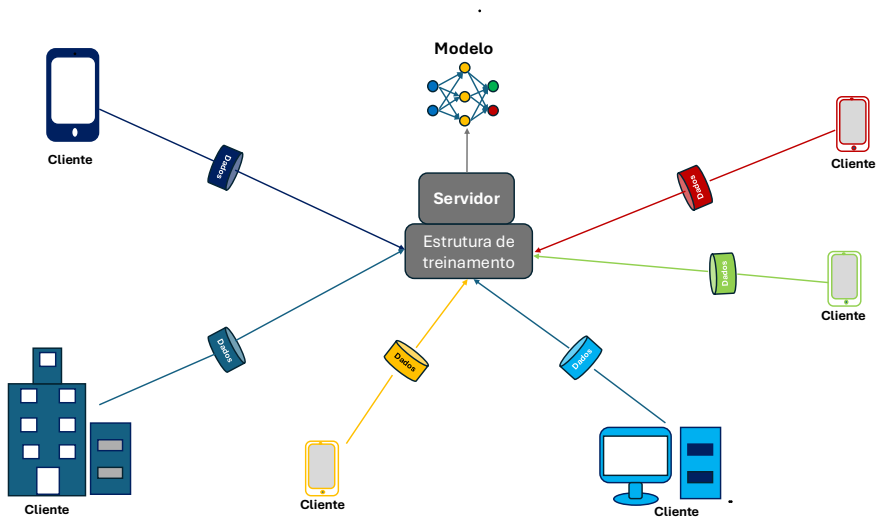

Figura 4 – Representação do funcionamento básico do ML centralizado. Os clientes compartilham seus dados com o servidor que através de uma estrutura de treinamento estabelece um modelo.

Para superar as limitações do ML centralizado, o Aprendizado Federado surge como uma abordagem inovadora no campo do Aprendizado de Máquina, transformando a forma como os modelos são treinados. Sua ideia central é o compartilhamento dos parâmetros (ou gradientes) dos modelos locais, em vez dos dados. A Figura 5 apresenta o funcionamento básico do FL. Os clientes realizam o treinamento utilizando seus dados locais e, em seguida, enviam os parâmetros do modelo para o servidor. O servidor utiliza um algoritmo de agregação para combinar esses parâmetros e gerar um modelo global. Esse

modelo global é, então, enviado de volta aos clientes, que o utilizam como ponto de partida para o treinamento local. Esse processo é repetido de forma iterativa até a convergência do modelo ou um critério de parada seja alcançado (MCMAHAN et al., 2017).

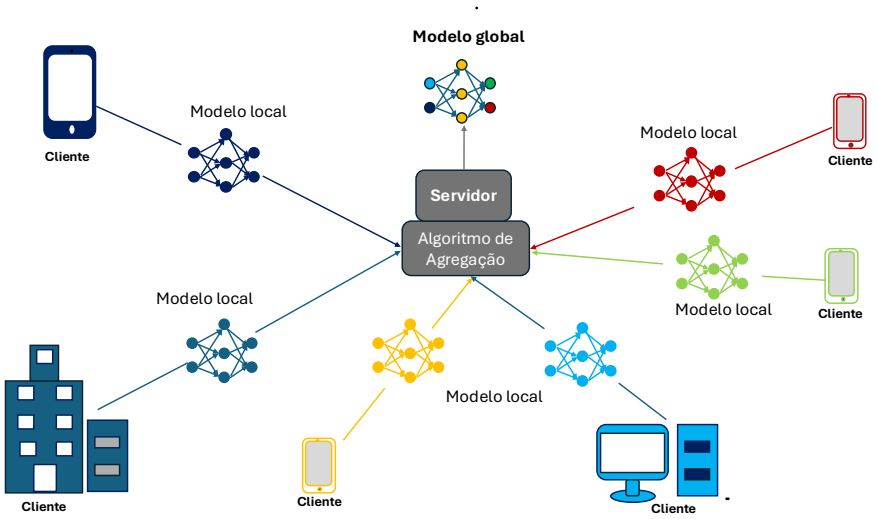

Figura 5 – Representação do funcionamento básico de FL. Os clientes treinam seus modelos utilizando dados locais, e os parâmetros gerados são enviados para o servidor. O servidor aplica um algoritmo de agregação para combinar os parâmetros locais dos clientes e gerar um modelo global.

A proteção de dados em ambientes de aprendizado colaborativo descentralizados é essencial em instituições onde as informações são sigilosas (LIU et al., 2020). Nesse contexto, o FL demonstra grande aplicabilidade em setores como indústria, automotivo, saúde pública, telecomunicações, entre outros. Além disso, o FL oferece escalabilidade, pois os dispositivos de borda não precisam transmitir os dados brutos, o que reduz os custos de comunicação, especialmente em redes com restrições de largura de banda (MCMAHAN et al., 2017; LI et al., 2021).

2.3.1 Tipos de FL

O FL pode ser classificado em três categorias, de acordo com a forma como o conjunto de dados é particionado Yang et al. (2019): FL Horizontal, FL Vertical e FL de Transferência de Conhecimento. A Figura 6 exemplifica graficamente esta categorizações. A matriz D_k representa os dados do cliente k, onde cada linha representa uma amostra, e cada coluna uma característica. Especificamente, para problemas com rotulação, X é o espaço de características, Y é o espaço de rótulos e I serve para denotar o espaço de identificação ID de amostra. Desta forma, o conjunto completo de dados é denominado como (I, X, Y).

O Aprendizado Federado Horizontal, ver Figura 6(a), é um método de aprendizado federado em que os conjuntos de dados compartilham o mesmo espaço de características

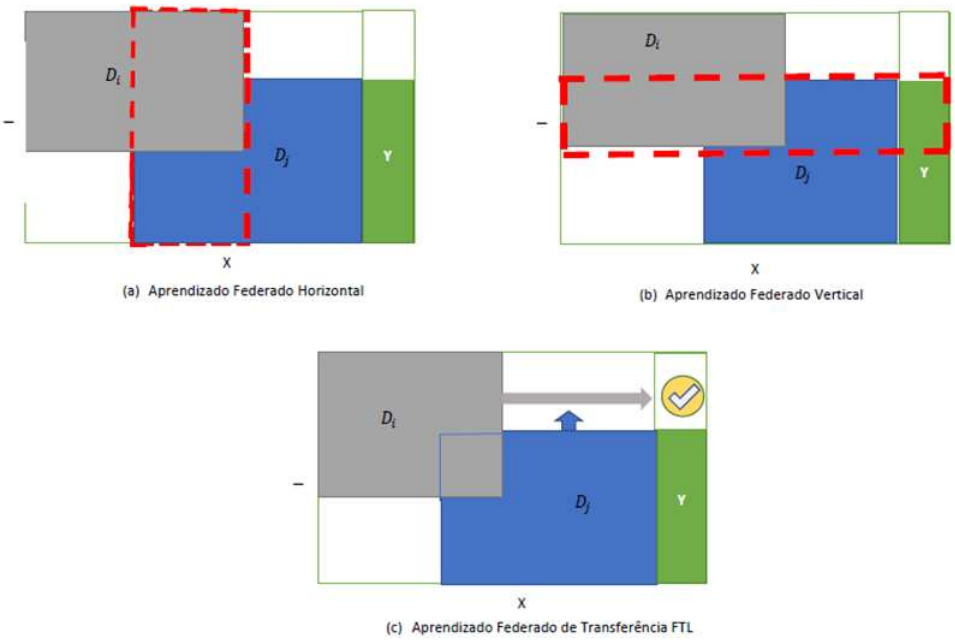

Figura 6 – (a) FL Horizontal, (b) FL Vertical e (c) FTL.

X, mas diferem nas amostras I, que é representada pela Equação (2.16). Essa técnica é usada em situações em que os dados de várias partes são semelhantes, mas as amostras podem diferir significativamente. Enquanto os dados brutos permanecem localizados e privados para cada participante, uma atualização de modelos de aprendizado de máquina requer colaboração. Ao permitir que várias partes trabalhem para construir um modelo global sem a necessidade de compartilhar dados brutos uns com os outros, este método garante a privacidade dos dados. Nessa abordagem as características X e os rótulos Y na aprendizagem federada horizontal são consistentes entre os vários participantes, mas as amostras individuais I podem ser únicas para cada participante. Quando diferentes entidades têm dados semelhantes, mas não desejam compartilhar detalhes, isso é especialmente útil (YANG et al., 2019).

$$X_i = X_j, \quad Y_i = Y_j, \quad I_i \neq I_j, \quad \forall D_i, D_j, i \neq j \tag{2.16}$$

Quando dois conjuntos de dados compartilham o mesmo espaço de ID de amostra, mas diferem no espaço de características X e/ou Y, então é o Aprendizado Federado Vertical, Equação (2.17), também conhecido como Aprendizado Federado baseado em características, é aplicável. A Figura 6(b) ilustra este tipo de FL, em que os cliente possuem a mesma quantidade de amostras mas com atributos diferentes (YANG et al., 2019).

$$X_i \neq X_j, \quad Y_i \neq Y_j, \quad I_i = I_j, \quad \forall D_i, D_j, i \neq j \tag{2.17}$$

O FTL apresentado na Figura 6(c), refere-se ao Aprendizado Federado de Transferência. O conceito de aprendizado federado se aplica a situações em que os conjuntos de

dados de treinamento diferem em amostra e espaço de características, Equação (2.18).

$$X_i \neq X_j, \quad Y_i \neq Y_j, \quad I_i \neq I_j, \quad \forall D_i, D_j, i \neq j \tag{2.18}$$

Técnicas de transferência de aprendizado são usadas para superar as diferenças de conjuntos de dados. Isso inclui aprender uma representação comum entre espaços de características usando conjuntos limitados de amostras comuns. Após isso, a representação comum é usada para fazer previsões em amostras com características de apenas um lado (YANG et al., 2019).

Baseado em arquitetura do sistema, FL pode ser Centralizado, Figura 5, ou Descentralizado, Figura 7, (MANZOOR et al., 2024). No FL descentralizado os clientes estabelecem redes de comunicação entre si, em uma configuração ponto-a-ponto (ISSA et al., 2023). Esta abordagem de FL consegue reduzir a quantidade de informações transmitidas e também melhora a eficiência de comunicação, já que cada cliente atualiza o seu modelo tendo em conta os modelos do vizinho imediato (MANZOOR et al., 2024b; YUAN et al., 2024).

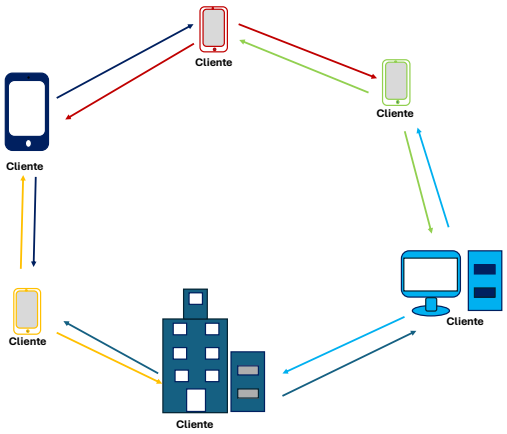

Figura 7 – FL descentralizado, onde os clientes se comunicam entre si sem a necessidade de um servidor central.

Quanto às estratégias operacionais o FL pode ser classificado em *Cross-Silo* e *Cross-Device* (MANZOOR et al., 2024b). *Cross-Silo* são operações de FL que envolvem pequenos grupos de repositórios com grandes conjuntos de dados, colaborando para o treinamento de um modelo, mantendo a privacidade e segurança dos dados (MÜLLER; BODENDORF, 2023; HUANG et al., 2023). Enquanto que, *Cross-Device* é a estratégia operacional de FL envolvendo dispositivos com pequenos conjunto de dados, capacidade de processamento relativamente baixa, e variações na disponibilidade dos dispositivos (DAI; MENG, 2023; WANG et al., 2023).

2.3.2 Tipos de Ataques em FL

Manzoor et al. (2024b) categorizam três tipos de ataques em FL. **Ataque contra os dados** busca corromper os dados de treinamento local, a fim de prejudicar todo o processo de aprendizado, esses ataques podem ser inserção de dados falsos ou modificações dos dados de treinamento (LV et al., 2022; LIU; XU; WANG, 2022). **Ataques contra o modelo** buscam degradar o modelo global, isto é, clientes maliciosos introduzem alterações nas suas atualizações para afetar negativamente o modelo global (MANZOOR et al., 2024a; MANZOOR et al., 2023). E por fim, **ataques contra a privacidade** que buscam violar a confidencialidade dos dados dos clientes, isto é, os invasores buscam inferir detalhes dos dados através da comunicação das atualizações ou do próprio modelo global (MOHAMMADI et al., 2024; YUAN et al., 2021; YANG et al., 2023).

Como existe diversos tipos de ataques em FL, também existe um número considerável de técnicas e estratégias para suprimir ou até conter esses ataques. Por exemplo, para suprimir ataques contra os dados pode-se utilizar técnicas robustas de validação de dados (JEBREEL; DOMINGO-FERRER, 2023). Para fazer frente a ataques contra os modelos, pode-se utilizar técnicas de regularização, algoritmos de agregação mais seguros, entre outros (WAINAKH et al., 2022). E técnicas como Privacidade Diferencial (WEI et al., 2020) e Criptografia Homomórfica (AZIZ et al., 2023) podem proteger contra ataque à privacidade.

2.3.3 Algoritmos de Agregação

A técnica de agregação chamada de Média Federada (FedAvg, do inglês *Federated Averaging*), foi a primeira abordagem do Aprendizado Federado, que serve também como o algoritmo referência que usa o sistema básico do funcionamento de FL (MCMAHAN et al., 2017). O FedAvg executa aprendizado colaborativo em rodadas síncronas usando uma arquitetura cliente-servidor. No início de cada rodada, alguns dos clientes recebem os parâmetros atuais do modelo atualizado do servidor (ou agregador). Como explicado anteriormente, o modelo é treinado localmente por cada cliente C_k usando seus próprios dados privados D_k. Em seguida, eles enviam de volta os parâmetros do modelo atualizados para o servidor. O servidor coleta essas atualizações e as combina usando uma **média ponderada** baseada na quantidade de dados locais de cada cliente. O modelo global recebe atualizações combinadas como um "pseudo-gradiente"(REDDI et al., 2020). Apesar de fornecer resultados empíricos sólidos em configurações com variáveis aleatórias independentes e identicamente distribuídas (*IID*) e classes equilibradas, o desempenho do FedAvg cai quando não for o caso (ZHAO et al., 2018).

Em cada rodada do FedAvg, o Servidor envia os parâmetros globais para o grupo S_t de clientes selecionados. Em seguida, os clientes utilizam a Equação (2.19) para

atualizar o parâmetros globais com base em seus dados locais. Os parâmetros atualizados são enviados de volta ao Servidor, que utiliza a Equação (2.20) para calcular a média ponderada das contribuições de todos os parâmetros locais recebidos, gerando assim novos parâmetros globais (modelo).

$$\theta_{ki} = \theta_i - \eta \nabla L(\theta_i; b_j) \tag{2.19}$$

e

$$\theta_i = \sum_{k=1}^{K_i} \frac{n_k}{n_{S_i}} \theta_{ki}, \tag{2.20}$$

onde η é a taxa de aprendizado, b_j é o j-ésimo mini-lote b, k é o k-ésimo Cliente na i-ésima iteração, θ são os parâmetros ou modelos (neste caso pesos e vieses), n_k é o número de amostras do k-ésimo cliente, e n_i é o número total das amostras dos clientes selecionados pelo Servidor na rodada i e K_i é o número total de clientes ativos na rodada i.

Tabela 1 – Resumo das abordagens de Aprendizado Federado (FL)

Autor/es (ano)	Algoritmo/s	Síntese de abordagens
Konečnỳ et al. (2016)	FL	Conceitualização e aplicações de Aprendizado Federado FL.
McMahan et al. (2017)	FedAvg	Aprendizado colaborativo em rodadas síncronas sob arquitetura Cliente-Servidor, onde o último faz atualizações mediante uma média ponderada.
Li et al. (2020b)	FedProx	Uma melhora de FedAvg, que procura colmatar as limitações apresentadas por FedAvg quando os dados são não IID.
Karimireddy et al. (2020)	SCAFFOLD	Aprimoramento dos métodos anteriores que procura acelerar a convergência e diminuir "desvio do cliente", e assim corrigir as atualizações locais.
Wang et al. (2020)	FedNova	Tenta otimizar o tempo da convergência, recursos de comunicação e memória, as atualizações são as versões reescaladas das mudanças locais.
Varno et al. (2022)	AdaBest	Procura estimar "desvio do cliente"com menos custos computacionais de FL em grande escala.

Surge o FedProx (LI et al., 2020b), como uma melhoria de FedAvg, que procura garantir convergência por meio da aprendizagem com dados não IID (heterogeneidade estatística) e da heterogeneidade de sistemas. Este algoritmo permite que cada dispositivo participante (cliente) realize uma quantidade variável de trabalho. Isto é, através de uma regularização, aplicada na função de custo, que minimiza tanto o erro quanto a distância entre os parâmetros do modelo local em relação ao parâmetros do modelo global, o que reduz as variações dos modelos locais e ajuda a estabilizar o processo de agregação no servidor. Desse modo, faz com que o treinamento de FedProx seja tolerante às heterogeneidades.

O SCAFFOLD (KARIMIREDDY et al., 2020) foi apresentado como método de acelerar a convergência e diminuir o chamado "*drift* do cliente"(mudanças graduais e contínuas nas características ou distribuições dos dados de um Cliente ao longo do tempo) nas atualizações locais. Como abordado anteriormente, em treinamentos federados com dados não IID os modelos locais dos clientes podem desestabilizar os modelos globais com o "*drift* do cliente". O SCAFFOLD possui controladores de variância que atuam para atenuar esses desvios. Diferentemente do FedProx, que utiliza regularização proximal para penalizar mudanças abruptas nos modelos locais, o SCAFFOLD trabalha de maneira mais direta na correção da direção dos gradientes, o que reduz a necessidade de ajustes excessivos em cada cliente.

Um algoritmo adaptativo introduzido pelo Varno et al. (2022), foi o AdaBest, que estima o "*drift* do cliente"com menos armazenamento e largura de banda de comunicação e menos custos computacionais. Além disso, aumenta a estabilidade ao restringir o padrão de estimativas de derivação dos clientes, tornando-o mais adequado para aprendizado federado em grande escala.

O FedNova, (do inglês *Federated Normalized Averaging*) (WANG et al., 2020), fornece um framework abrangente para as análises da convergência de algoritmos heterogêneos de otimização federada. Os autores quiseram entender a aceleração e a desaceleração da convergência causadas pela inconsistência do objetivo. O FedNova também agrega um método de média normalizada para eliminar a inconsistência do objetivo e garantir uma rápida convergência do erro. Os autores, conjecturaram que métodos como FedProx e SCAFFOLD, acima apresentados, só conseguem até certo ponto reduzir (não eliminar) a inconsistência do objetivo. No entanto, esses métodos podem levar mais tempo para convergir ou exigir mais recursos de comunicação e memória. Em vez de mudanças locais, as atualizações localmente normalizadas no FedNova, são as médias.

2.4 Trabalhos Relacionados

Muitos trabalhos científicos nos últimos anos evidenciaram a eficiência da utilização de RNAs em diferentes tipos de conjuntos de dados (OJHA; ABRAHAM; SNÁŠEL, 2017). Entretanto, também surgiram novas demandas relacionadas ao treinamento de RNAs, e uma delas, é a própria segurança e privacidade dos dados utilizados do treinamento (AUGENSTEIN et al., 2019).

Uma metodologia que tem demonstrado ótimos resultados em maximizar a eficiência dos modelos treinados é a utilização de algoritmos híbridos (JUANG, 2004; YADAV et al., 2020). Quanto à privacidade, FL tem se mostrado útil em termos da garantia mínima de privacidade e segurança dos dados (LI et al., 2020a).

2.4.1 Otimização Híbrida com PSO e SGD

Yadav et al. (2020) apresentaram um novo algoritmo de treinamento híbrido que combinava PSO-GA com o otimizador Adam. Os autores comparam o novo algoritmo ao algoritmo convencional de backpropagation com Gradiente Descendente e Otimização de Adam no treinamento de Redes Neurais Artificiais (YADAV et al., 2020).

Phong, Santos e Ribeiro (2022) produziram o algoritmo PSO híbrido com SGD para o treinamento de CNNs para otimizar as taxas de aprendizagem (l*earning rate*). As dinâmicas das partículas do PSO correspondem às configurações dos hiper-parâmetros da Rede Neural, que são ajustados através do treinamento em duas fases, que são, a fase regular e a fase colaborativa.

No trabalho de Kaya, Yılmaz e Aşar (2023), abordou-se a importância do diagnóstico precoce de sepse para reduzir o risco de morte do paciente. Destaca-se a limitação de algoritmos baseados em gradiente, com sua propensão a ficarem presos em mínimos locais. Nesse contexto, o artigo propõe um novo algoritmo híbrido baseado em metaheurística, utilizando a Otimização por Enxame de Partículas (PSO) e o operador de busca mental do algoritmo de Busca Mental Humana (HMS). Para assim, otimizar os pesos de uma rede neural profunda (HMS-PSO-DNN) no contexto do diagnóstico precoce de sepse.

Os estudos acima citados destacaram a eficácia de abordagens híbridas, combinando PSO e outras técnicas de otimização baseadas no SGD para melhorar o desempenho dos modelos treinados por RNAs em diversas aplicações. Buscando com isso, superar as limitações de métodos isolados. Ou seja, esses métodos e estratégias, buscam explorar as vantagens complementares dos algoritmos envolvidos na hibridização. A Tabela 2 descreve resumidamente as abordagens híbridas de PSO supra mencionadas.

Tabela 2 – PSO Híbrida

Autor/es (ano)	Algoritmo/s	Otimização	Tipo de Rede
Yadav et al. (2020)	PSO-GA-Adam	Pesos e Vieses	ANN/DNN
Phong, Santos e Ribeiro (2022)	PSO-CNN	Taxa de Aprendizagem	CNN
Kaya, Yılmaz e Aşar (2023)	HMS-PSO-DNN	Pesos e Vieses	DNN

2.4.2 Aprendizado Federado com PSO

O FedPSO (PARK; SUH; LEE, 2021) tem como foco principal diminuir a quantidade de parâmetros transmitidos do cliente para o servidor, o que é alcançado por meio de uma avaliação dos valores dos erros ou acurácias obtidos nos treinamentos locais dos clientes. Apenas os parâmetros do cliente com melhor acurácia ou menor erro são enviados para o servidor, que por sua vez, redistribui esses parâmetros como atualização da rodada

seguinte para os clientes. Essa abordagem reduz a quantidade de parâmetros totais que são enviados para o servidor, o que diminui o custo de comunicação comparado ao FedAvg. Entretanto, não proporciona ganhos significativos de desempenho, uma vez que os clientes realizam treinamentos locais usando SGD da mesma forma que FedAvg.

Victor et al. (2023) apresentaram o algoritmo FL-PSO. Demonstraram que a aplicação de FL em conjunto com PSO obteve uma melhor precisão na previsão precoce de Acidente Vascular Cerebral (AVC) em comparado aos outros métodos usados. Nessa abordagem os autores utilizaram PSO como otimizador dos hiper-parâmetros do sistema de comunicação das redes de FL.

O trabalho de Bakir, Samrouth e Samrouth (2023) é um estudo que propõe o uso do FedPSO-GA para otimizar o aprendizado federado na previsão do consumo de energia em edifícios inteligentes equipados com sensores e dispositivos IoT. A abordagem visa superar restrições de comunicação e privacidade de dados. O FedPSO-GA combina o algoritmo genético (GA) com o método FedPSO para ajustar os parâmetros do modelo global, otimizando assim o desempenho dos modelos de FL. Os experimentos demonstraram que o método proposto, no referido trabalho, melhora a previsão de consumo de energia em comparação com FedAvg e FedPSO, reduzindo o Erro Quadrático Médio (RMSE).

O FedPSO e o FedPSO-GA são métodos que conseguem reduzir significativamente os custos da comunicação cliente-servidor, entretanto possuem uma eficiência relativamente baixa em termos de treinamento local dos clientes (PARK; SUH; LEE, 2021; BAKIR; SAMROUTH; SAMROUTH, 2023). Enquanto que o método FL-PSO sofre com níveis altos de custo de comunicação, devido a quantidade de parâmetros que são transferidos entre cliente e servidor e vice versa (VICTOR et al., 2023).

A Tabela 3 descreve em resumo as abordagens para os algoritmos, relativamente aos trabalhos apresentados nesta subseção. Mostrando o tipo de algoritmo de agregação, também fazendo uma síntese da proposta do trabalho.

Tabela 3 – Resumo das abordagens de Aprendizado Federado (FL)

Autor/es (ano)	Algoritmo/s	Síntese de abordagens
Park, Suh e Lee (2021)	FedPSO	Utilização de FedPSO como alternativa para FedAvg, os resultados obtidos evidenciaram melhoras significativas comparado ao FedAvg.
Victor et al. (2023)	FL-PSO	PSO é utilizado como otimizador de hiperparâmetros da estrutura de FL.
Bakir, Samrouth e Samrouth (2023)	FedPSO-GA	Uma abordagem que melhora FedPSO acrescentando GA no processo de aprendizagem.

3 APRENDIZADO FEDERADO COM PSO-SGD

Este capítulo apresenta a metodologia proposta no presente trabalho, isto é, uma técnica de agregação em FL. Sendo assim, este algoritmo engloba a técnica híbrida proposta para o treinamento local dos clientes, e a transmissão de valores de erros do cliente para o servidor.

O algoritmo proposto, que foi denominado de FLPSO-SGD, é um método colaborativo de treinamento para otimização de parâmetros de RNAs com dados distribuídos. Para um melhor entendimento do método de Aprendizado Federado proposto, que é baseado em Otimização por Enxame de Partículas hibridizado com o gradiente Descendente Estocástico, o algoritmo será dividido em duas partes, a primeira parte mostrando o funcionamento da estratégia do servidor. Enquanto que a segunda parte apresentará a estratégia de treinamento dos clientes, que utiliza a abordagem híbrida sugerida (PSO-SGD).

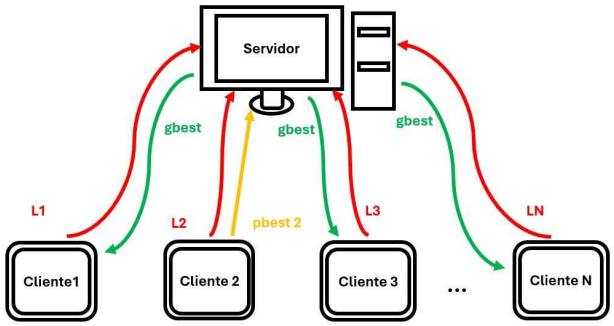

Figura 8 – FLPSO-SGD. Os clientes enviam os erros L_k (setas vermelhas) e exclusivamente o de melhor desempenho envia os parâmetros (seta amarela). O servidor atualiza esses parâmetros para os clientes (setas verdes).

A Figura 8 mostra N dispositivos (clientes), onde cada um treina localmente o seu modelo, utilizando o seu próprio conjunto de dados diferentes. Depois do treinamento local, cada cliente envia o seu valor de erro para o servidor, que compara os valores recebidos, e solicita os parâmetros do cliente vencedor (cliente com menor erro). Desse modo, o algoritmo mantém a privacidade de dados dos clientes, enquanto diminui consideravelmente a quantidade dos parâmetros que são transmitidos dos clientes para o servidor.

Quando os clientes enviam seus erros para o servidor, esse processo não só diminui o custo de comunicação (ao diminuir a quantidade de parâmetros transmitidos), mas também acaba garantindo uma certa segurança ao sistema inteiro. Isto é, por servidor receber só os parâmetros do cliente vencedor, limita os efeitos de possíveis ataques de

inferência com objetivo de violar a confidencialidade dos clientes (YUAN et al., 2021; YANG et al., 2023). Da mesma forma, quando o servidor envia o modelo global *gbest* para um grupo de clientes selecionados, as alterações que possam advir de ataques contra o modelo são atenuadas devido ao componente social e componente da inércia do PSO (MANZOOR et al., 2024a; MANZOOR et al., 2023).

O restante deste capítulo explica em detalhe como acontece o treinamento em FL usando FLPSO-SGD através do gerenciamento do servidor (Lado do Servidor). E também, através do treinamento local dos clientes (Lado do Cliente).

3.1 Lado do Servidor

O processo de gerenciamento do treinamento FL do método proposto se encontra resumido no Algoritmo 2. Para cada rodada de comunicação definida (linha 1), o servidor recebe os erros L_k enviados pelos N clientes (setas vermelhas na Figura 8) e avalia, qual é o cliente com menor erro (linha 2) utilizando as Equações 3.1 e 3.2 .

$$L_{\min} = \min(L_1, L_2, \ldots, L_N) \tag{3.1}$$

$$k^* = \arg\min_k L_k \tag{3.2}$$

Posteriormente, o servidor solicita o *pbest* (seta amarela na Figura 8) do cliente com o melhor valor *fitness* (menor erro), e em seguida atribui estes valores como *gbest* (linha 3). Finalmente, um subconjunto aleatório S_t de clientes é selecionado para receber *gbest* do servidor (setas verdes na Figura 8), e atualizar os seus parâmetros aplicando a função de atualização do cliente (linha 5-7). Este processo ocorre paralelamente para todos os clientes pertencentes a subconjunto S_t. A atualização do cliente (linha 6) se refere a aplicação do algoritmo híbrido utilizado para treinamento local (pseudocódigos do Algoritmo 3) que será explicado na próxima subseção.

Após o servidor fornecer o *gbest* para o grupo de clientes selecionados (S_t), inicia-se o treinamento local de cada cliente. Durante esse processo, novos conjuntos de parâmetros são iterativamente atualizados e refinados. Em seguida, os erros obtidos são enviados de volta ao servidor. Esse ciclo de atualização e comunicação é repetido até que o número predeterminado de rodadas seja alcançado.

3.2 Lado do Cliente

Os conceitos básicos de Otimização por Enxame de Partículas foram abordados no capítulo anterior. O algoritmo híbrido apresentado aqui é uma modificação da Equação

Algorithm 2 FLPSO-SGD

Para cada rodada $t = 1, 2, 3...$ **faça**
    $L_{min} \longleftarrow \min(L_1, L_2, \ldots, L_N)$
    Solicita o *pbest* do Cliente k^*
    servidor escolhe subconjunto S_t de clientes
    para cada cliente $k \in S_t$ em paralelo
        $L_k \longleftarrow$ Atualização_Cliente(k, g_{best})
    Fim Para
Fim Para

(2.10), que é a equação de atualização de velocidade de PSO, isto é, adicionando um componente de gradiente descendente estocástico. Essa proposta é representada pela Equação (3.3).

$$v_{k(i+1)} = wv_{ki} + c_1 r_{1ki}(pbest_{ki} - \theta_{ki}) + c_2 r_{2ki}(gbest_i - \theta_{ki}) - \eta \nabla L(\theta_{ki}; D_j) \tag{3.3}$$

onde θ_{ki} representa uma configuração dos parâmetros (pesos e bias) da RNA, que se atualiza minimizando a função do erro L, $pbest_k$ são os parâmetros ótimos pessoais do cliente e $gbest_i$ parâmetros ótimos globais (geralmente fornecidos pelo servidor). Além disso, $\nabla L(\theta_{ki}; D_j)$ representa o gradiente da função de custo em relação ao parâmetros aplicados ao mini-lote D_j, o índice k representa $k-$ésima partícula e o índice i representa $i-$ésima iteração. A atualização da posição da partícula, que corresponde a parâmetros atualizados da rede neural, é feita através da Equação (2.11).

Como referido anteriormente, a versão *gBest PSO* na Equação (3.3) consegue atenuar o modelo local dos clientes, usando as componentes do inércia (v_i e w_i) e o fator social (c_1 e $gbest_i$), isto evita que os os modelos locais consigam fazer alterações abruptas e criando desvios, sobretudo em treinamentos com dados não IID. Também foi abordado os ataques em FL, ou seja, Manzoor et al. (2024) categorizaram os tipos de ataques em aprendizado federado em: ataque aos dados, ataque ao modelo e ataque à privacidade. Ataque contra os dados, pode cingir-se na corrupção dos dados locais dos clientes com objetivo de afetar o processo do treinamento federado e comprometendo o modelo global. O ataque ao modelo, pode ser a manipulação intencional das atualizações do modelos locais dos clientes, com objetivo de corromper o modelo global. E por último, ataque à privacidade que visa reverter a natureza preservadora de privacidade de FL para obter dados sensíveis dos clientes. Utilizando a Equação (3.3) para atualização da velocidade do cliente, pode-se proteger a privacidade dos dados, e simultaneamente atenuar a influência de um modelo global alterado. Isso torna, os modelos globais e locais mais resilientes aos ataques devido ao peso da inércia w e modelo global atualizado $gbest$ e seu coeficiente c_2.

A Figura 9 ilustra a dinâmica do algoritmo híbrido proposto para o treinamento local. Nesta figura, a soma vetorial v_{i+1}, representada pelo vetor azul, corresponde a

uma atualização da velocidade utilizando o algoritmo do PSO original (KENNEDY; EBERHART, 1995). A modificação proposta neste trabalho, consiste na agregação do componente do SGD com uma taxa de aprendizado, resultando na atualização da velocidade representada na Equação (3.3), mostrado pelo vetor preto. Em seguida, os parâmetros são atualizados para x_{i+1}, somando x_i e v_{i+1}, como mostra a Equação (2.11).

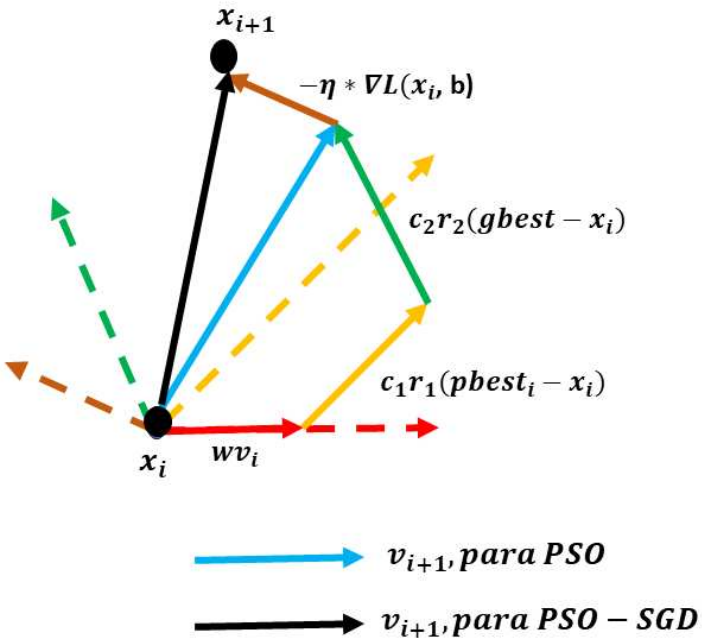

Figura 9 – Representação vetorial do algoritmo híbrido PSO-SGD proposto para a otimização dos parâmetros locais de RNAs.

O Algoritmo 3 descreve o processo proposto para o treinamento local dos clientes. Primeiramente, os dados do cliente são divididos em lotes pequenos e os parâmetros do cliente k são inicializados. Se o *gbest* inicial do cliente for um vetor nulo (sem fornecimento inicial do *gbest* pelo servidor) os parâmetros de x_k são inicializados aleatoriamente e a velocidade inicial da partícula (cliente) é zero. Isso faz com que, só o último termo (SGD) da Equação (3.3) inicializa a primeira época do treinamento do cliente. Os valores de erros são comparados, caso a condição de atualização (linha 13) seja cumprida o *pbest* é atualizado (linha 14), este processo é repetido de acordo com a quantidade de épocas definidas para o treinamento local e quantidades de mini-lotes que o cliente possui. Após atingir o número de épocas de treinamento especificadas o último erro obtido neste treinamento local é enviado ao servidor (linha 18), para ser avaliado junto aos erros de outros clientes.

As inicializações de velocidades e escolhas dos pesos da inércia e coeficientes de aceleração sempre foram muitos importante nas dinâmicas do PSO (SHI; EBERHART, 1998). Na Equação (3.3), as inicializações podem ser aleatórias, as escolhas dos coeficientes de aceleração podem ser puramente empíricos e o peso da inércia utilizando a Equação (2.12).

Algorithm 3 Função de Atualização do Cliente

Função Atualização_Cliente $(k, gbest)$
    Divisão dos dados do cliente em B lotes
    Se $gbest = 0$, **Então**
        inicializa θ_k aleatoriamente
        $v_k \longleftarrow$ vetor nulo
        $p_{\text{best}} \longleftarrow \theta_k$
        $g_{\text{best}} \longleftarrow \theta_k$
    Fim Se
    Para cada época do cliente $i = 1, 2, ...$ **faça**
        Para cada lote $b \in B$ **faça**
            execução da equação (3.3)
            $\theta_{i+1} \longleftarrow \theta_i + v_{i+1}$
            Se $L(D_k, \theta_{i+1}) < L(pbest)$, **Então**
                Atualiza $pbest \longleftarrow \theta_{i+1}$
            Fim Se
        Fim Para
    Fim Para
    retorna $L(D_k, \theta_{i+1})$
Fim da Função

4 EXPERIMENTOS E RESULTADOS

Os experimentos foram realizados em duas fases, a primeira fase foi realizada como a validação do método de otimização híbrida PSO-SGD e a segunda fase foi a aplicação do algoritmo de aprendizado federado proposto FLPSO-SGD.

4.1 Base de Dados

Para a primeira fase dos experimentos, foram utilizados 7 diferentes conjuntos de dados *(datasets)*, correspondentes aos problemas clássicos de classificação, que foram obtidos através do repositório para Aprendizado de Máquina da *UC Irvine* ou *(UC Irvine Machine Learning Repository)*. A Tabela 4 mostra os detalhes principais dos conjuntos de dados (número de instâncias, número de atributos e número de classes).

Tabela 4 – Detalhes de conjunto de dados clássicos usados na 1ª fase dos experimentos

Conjunto de dados	Instâncias	Atributos	Classes
Iris	150	4	3
Breast Cancer	569	31	2
Glass	214	9	6
Banknote	1372	5	2
Ionosphere	351	34	2
Sonar	208	60	2
Wine	178	13	3

O segundo tipo de experimentos realizados na primeira fase, com a utilização de arquiteturas CNNs. Nesses experimentos foram utilizados o conjunto de dados de Visão Computacional CIFAR-10 (*Canadian Institute for Advanced Research, 10 classes*), que pertence ao conjunto de dados conhecidos como *"Tiny Images"*. O conjunto de dados contém uma grande coleção de imagens variadas, de baixa resolução que frequentemente são usadas em problemas de Aprendizado de Máquina, sobretudo, no campo de Visão Computacional. CIFAR-10 possui 60.000 amostras de imagens coloridas de 32x32 pixels, que são: avião, automóvel, pássaro, gato, veado, cachorro, sapo, cavalo, navio e caminhão. Existem 6.000 imagens por classe, com 5000 imagens para o treinamento e 1000 imagens por classe para testar o modelo. A Figura 10 mostra algumas imagens aleatórias do conjunto de dados CIFAR-10.

Na segunda fase dos experimentos, também utilizou-se CIFAR-10 para aplicação do algoritmo de aprendizado federado desenvolvido FLPSO-SGD. Nessa fase, também foram realizados dois tipos de experimentos para diferentes arquiteturas CNNs. O primeiro

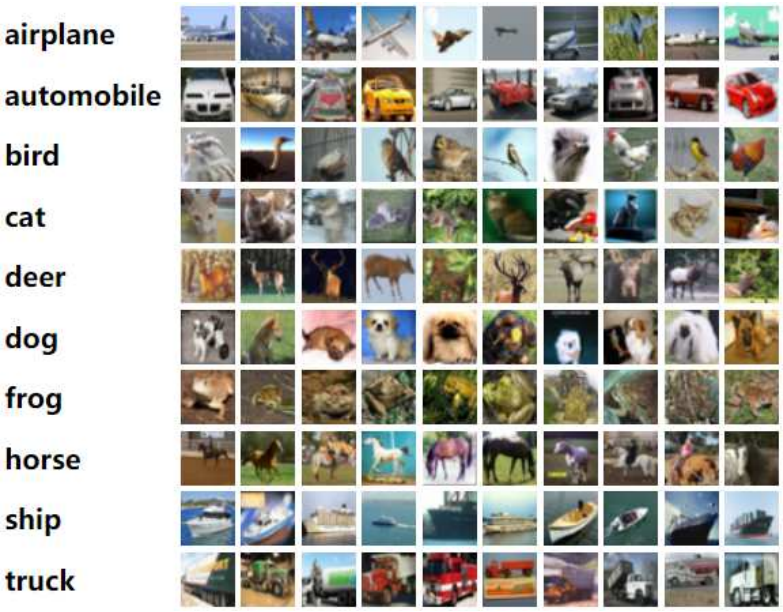

Figura 10 – Conjunto de algumas imagens aleatórias do CIFAR-10 (KRIZHEVSKY; HINTON et al., 2009).

tipo de experimentos foi realizado com dados completos do CIFAR-10 distribuídos para cada cliente, enquanto que o segundo tipo de experimentos nessa fase foi realizado com dados particionados do CIFAR-10, simulando a distribuição dos dados não IID.

4.2 Configurações Experimentais

Para os experimentos utilizando conjuntos de dados clássico de ML, configurou-se uma arquitetura de RNA com uma camada oculta contendo 20 neurônios. As camadas de entradas e saídas foram configuradas de acordo com os atributos e classe dos problemas, respectivamente. Na camada oculta foram utilizadas ReLU como a função de ativação, na camada de saída foi usada sigmoide como classificador para os problemas de classificação binária. Assim, para problemas de classificação multi-classe utilizou-se Softmax como classificador, usando a Entropia Cruzada como função de custo. Os hiper-parâmetros pertinentes foram configurados com os seguintes valores: A taxa de aprendizagem foi de 0,01 para todos os conjuntos de dados menos para *Glass* que foi de 0,001. Essa mudança no valor da taxa de aprendizado, se deve à complexidade do problema *Glass* (problema com maior número de classes na primeira fase dos experimentos). E também, devido a eficiência experimental empírica, foi adotado o valor de 0,001 para o referido problema. Além disso, a população inicial de PSO para todos os conjuntos de dados foi de 25 partículas, com os parâmetros $c_1 = 0,8$, $c_2 = 0,5$, $w = 0,9$ e com 1000 iterações como critério de parada. Para todos os conjuntos de dados utilizados, no primeiro tipo de experimentos desta fase experimental, utilizou-se 20% de dados para teste, os erros de validação foram obtidos

aplicando a validação cruzada com 5-folds. Isto é, o conjunto de treinamento foi subdividido em 5 partes, cada iteração usa 4 partes para o treinamento e 1 parte para validação, e depois das 1000 iterações, o erro de validação é obtido somando-se os erros médios dos folds obtidos em cada iteração e divide-se o resultado pelo número total de iterações. No primeiro tipo de experimento da primeira fase, os experimentos foram realizados em um computador pessoal (PC) com 32 GB de RAM instalada, e processador Intel(R) Core(TM) i5-10400F CPU @ 2.90GHz.

Nos experimentos onde se utilizou o conjunto de dados CIFAR-10 (segundo tipo de experimento da primeira fase e a segunda fase completa), onde se aplicou o algoritmo PSO-SGD ou FLPSO-SGD a este problema de Visão Computacional (CIFAR-10), através de arquiteturas CNNs. A taxa de aprendizagem foi determinada de modo empírico como 0.0001, com 30 épocas, 10 partículas (clientes), com os hiper-parâmetros $c_1 = 1{,}5$, $c_2 = 1{,}8$, e $w = 0{,}9$ decaindo linearmente conforme a época e iteração do cliente, até atingir o valor mínimo possível $w = 0{,}4$ como mostrado na Tabela 5. Mais à frente, as configurações das arquiteturas convolucionais são mostradas na Tabela 6. Os Experimentos computacionais relativos às CNNs, foram realizados no mesmo PC da fase anterior, em conjunto com a placa de vídeo GPU, NVIDIA GeForce RTX 3060, com 12 GB de VRAM, com 3584 núcleos CUDA de processamento.

Tabela 5 – Configurações dos Hiperparâmetros

Dados	Fase	Algoritmo	η	w	c_1	c_2	Nº Partíc.	Iter.	Iter. Clientes	RNA
Clássicos (UC Irvine)	1	PSO-SGD	0,01/0,001	0,9	0,8	0,5	25	1000	-	MLP Simples
CIFAR-10	1	PSO-SGD	0,0001	0,9 - 0,4	1,5	1,8	10	30	-	CNNs
CIFAR-10	2	FLPSO-SGD	0,0001	0,9 - 0,4	1,5	1,8	10	30	5	CNNs
CIFAR-10 (não IID)	2	FLPSO-SGD	0,0001	0,9 - 0,4	1,5	1,8	10	30	5	CNNs

A Tabela 5 faz um resumo dos principais elementos das configurações experimentais utilizadas no presente trabalho, onde η se refere a taxa de aprendizado, Nº partículas é o tamanho da população ou quantidade dos clientes, iterações são as épocas do treinamento ou rodadas do servidor e iterações clientes é a quantidade de épocas de treinamento local dos clientes.

As métricas de Erro faram calculadas utilizando a função de perda Entropia Cruzada Equação (4.1), que mede a diferença entre duas distribuições de probabilidades e a Acurácia é a quantidade de previsões corretas dividido por número total de exemplos foi calculada usando a Equação (4.2).

$$\text{Erro} = \boldsymbol{L} = -\frac{1}{N}\sum_{j=1}^{N} \log\left(\frac{\exp(z_{j,t_j})}{\sum_{l=1}^{C}\exp(z_{j,l})}\right) \tag{4.1}$$

$$\text{Acurácia} = \boldsymbol{Acc} \ = \frac{1}{N}\sum_{j=1}^{N} 1\left(\arg\max_{l} z_{j,l} = t_j\right) \tag{4.2}$$

Onde N é o número total de amostras, C é o número de classes, $z_{j,l}$ é o logit da amostra j para classe l, o t_j é o índice da classe verdadeira da amostra j. A função indicadora $1\left(\arg\max_l z_{j,l} = t_j\right)$ retorna 1 caso a classe prevista $\arg\max_l z_{j,l}$ for igual a classe real t_j e 0, caso contrário.

4.3 Arquiteturas CNNs Utilizadas

As configurações utilizadas para aplicações de CNNs são as mesmas tanto na primeira fase quanto na segunda fase, exceptuando que a segunda fase comporta o treinamento federado. As arquiteturas convolucionais utilizadas são: SqueezeNet (IANDOLA, 2016), MobileNetV2 (SANDLER et al., 2018), ResNet18 (HE et al., 2016) e AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012).

SqueezeNet, proposta em 2016, com intuito de reduzir a quantidade dos parâmetros da rede, enquanto mantém um desempenho consideravelmente boa em classificações de imagens. Isto é, essa rede tenta reduzir a quantidade dos parâmetros mantendo a acurácia o mais alto possível, para isso, utiliza as convoluções 1x1 (squeeze) seguida de convoluções 3x3 (expansão), permitindo assim uma rede compacta (IANDOLA, 2016).

MobileNetV2 é uma rede com objetivo de ter altos desempenhos nos dispositivos de borda, utilizando convoluções profundas separáveis e uma topologia de camada chamada *inverted residuals*. Com isso, procura reduzir as dimensionalidades das convoluções, assim, a rede consegue reduzir os parâmetros treináveis enquanto melhora o desempenho (SANDLER et al., 2018).

ResNet18, as arquiteturas ResNet (do inglês *Residual Networks*) introduziram o conceito de aprendizado residual, isto é, algumas conexões neurais permitem que as informações pulem algumas camadas e sejam adicionadas às saídas das camadas subsequentes. Permitindo assim, o treinamento de redes profundas sem o problema do desaparecimento de gradientes. ResNet18 é a versão da arquitetura ResNet que possui 18 camadas de parâmetros treináveis (HE et al., 2016).

AlexNet também foi utilizada, essa arquitetura foi proposta em 2012, quando ganhou o desafio ILSVRC (do inglês *ImageNet Large Scale Visual Recognition Challenge*) do referido ano. Esta rede constitui uma das primeiras aquiteturas CNN a conseguir uma avanço significativo no campo de Visão Computacional. Essa rede introduziu várias inovações, entre elas, a utilização da função ReLU (do inglês *Rectified Linear Unit*) para acelerar o treinamento, *Dropout* para reduzir *overfitting* e convoluções com grandes profundidades de rede. AlexNet possui 8 camadas treináveis, isto é, 5 camadas convolucionais e 3 camadas totalmente conectadas (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). Para facilitar a comparação e reprodutibilidade dos experimentos nesta dissertação, para todas as implementações envolvendo CNNs foi fixado um *Seed(123)*.

Tabela 6 – Configurações das Arquiteturas CNNs Utilizadas

Tipo de Camada	**SqueezeNet**	**MobileNetV2**	**ResNet18**	**AlexNet**
Conv2d	25	17	20	5
ReLU	30	19	21	7
MaxPool2d	3	0	1	3
AdaptiveAvgPool2d	1	1	1	1
Dropout	1	1	0	2
Linear	1	1	1	3
BatchNorm2d	0	16	9	0
BasicBlock	0	0	8	0
Fire Module	21	0	0	0
InvertedResidual	0	17	0	0
Total de Parâmetros	727.626	2.236.682	11.181.642	57.044.810
Tamanho de Parâmetros (MB)	2,78	8,50	42,70	217,61
Tamanho Estimado (MB)	54,55	161,97	106,01	226,55

A Tabela 6 faz um detalhamento das configurações das CNNs utilizadas no presente trabalho da seguinte forma: *Conv2d* é a camada convolucional, que aplica um filtro (ou kernel) sobre a entrada para extrair características locais. ReLU é a camada que comporta a função de ativação do mesmo nome, Equação (4.3), que substitui todos os valores negativos por zero como mostra a Figura 11 (NAIR; HINTON, 2010).

$$\text{ReLU}(x) = \max(0, x) \tag{4.3}$$

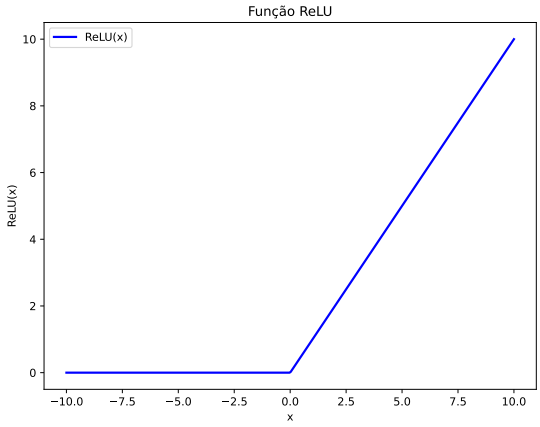

Figura 11 – Função ReLU.

As camadas *pooling* reduzem as dimensões das características mantendo as mais relevantes, *MaxPool2d* faz isso, escolhendo somente os valores máximos das características. Enquanto que *AdaptiveAvgPool2d* realiza essa tarefa de forma adaptativa fazendo a média dos valores. A Figura 12 mostra um exemplo de funcionamento da camada Convolucional e *Pooling* (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). *Dropout* é uma camada em que se aplica uma técnica de regularização para evitar *overfitting*. A camada Linear ou totalmente conectada, conecta cada neurônio a todos os neurônios da camada seguinte.

BatchNorm2d é a camada que faz a normalização para uma média zero e desvio-padrão unitário (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). *BasicBlock* é uma camada usada em arquiteturas *ResNet* e inclui as conexões residuais (HE et al., 2016). *Fire Module* ou módulo *Fire* é uma estrutura específica da arquitetura *SqueezeNet*, composta por uma camada squeeze (comprimida) e seguida de uma camada expandida (IANDOLA, 2016). A *InvertedResidual* é uma camada fundamental para *MobileNetV2*, pois inverte os blocos residuais convencionais, para reduzir a quantidade dos cálculos necessários e ao mesmo tempo manter a eficiência (SANDLER et al., 2018).

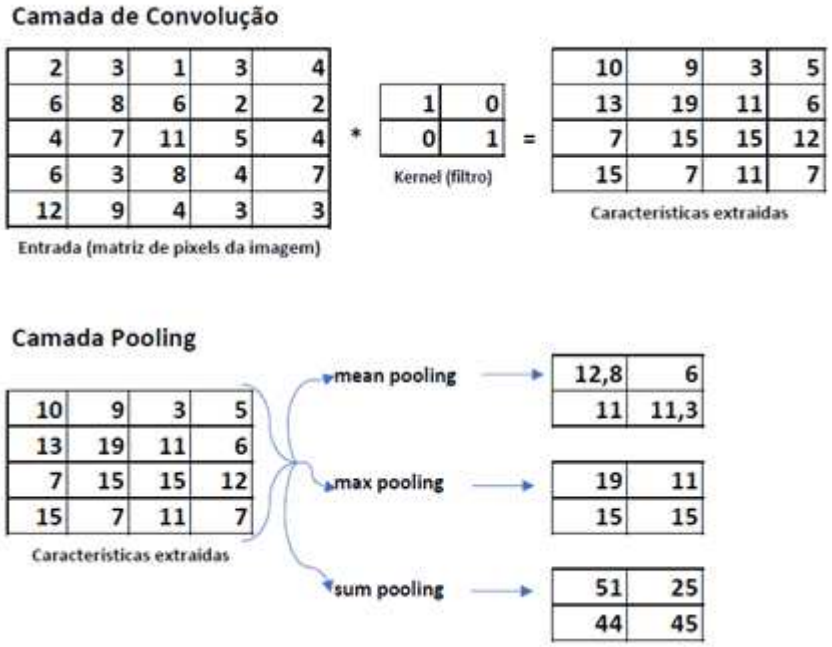

Figura 12 – Exemplo de uma camada convolucional e uma *pooling*.

Total de Parâmetros na Tabela 6 é a soma de todos os parâmetros treináveis da arquitetura, Tamanho de Parâmetros é a quantidade total de parâmetros em MB (usando float32 característico do *framework* Pytorch). E por fim, o Tamanho Estimado envolve o tamanho de parâmetros mais tamanho de entrada e tamanho de *forward/backward pass*. Os principais códigos implementados no presente trabalho estão disponíveis em repositória GitHub https://github.com/EliRockySanha/EliezerSanha.

4.4 Resultados e Discussões

Os resultados obtidos foram divididos em duas fases experimentais como descrito na Tabela 6. A Fase Experimental 1 contém os resultados obtidos da aplicação dos setes conjuntos de dados clássicos do ML mostrados na Tabela 4 e as quatro arquiteturas convolucionais ao CIFAR-10, para **PSO-SGD**. Enquanto que a Fase Experimental 2, são resultados relativos às aplicações de três arquiteturas das quatro arquiteturas convolucionais

(redução das arquiteturas aplicadas nessa fase devido ao custo computacional) descritas anteriormente ao **FLPSO-SGD**.

4.4.1 Fase Experimental 1

Os resultados obtidos no primeiro tipo de experimentos realizados na fase 1 estão dispostos na Tabela 7, sendo ***t(s)*** representando o valor de tempo de treinamento em segundos; ***L*** representa o erro do treinamento; e por fim ***Acc*** representando a acurácia do modelo.

Tabela 7 – Resultados dos sete conjuntos de dados da UC Irvine na fase 1

		Adagrad	RMSprop	Adam	PSO-SGD
Iris	***t(s)***	4,381 ±0,100	4,377 ±0,099	4,507 ±0,106	**4,338** ±0,080
	L	3,893E-01 ±0,239	1,294E-01 ±0,173	1,387E-01 ±0,219	**1,238E-05** ±1,50E-05
	Acc	0,864 ±0,158	0,916 ±0,126	0,929 ±0,120	**0,973** ±0,034
B. Cancer	***t(s)***	**4,366** ±0,071	4,420 ±0,058	4,564 ±0,070	4,426 ±0,066
	L	5,383E-02 ±0,005	1,574E-05 ±6,97E-06	3,748E-04 ±4,30E-04	**6,183E-06** ±7,02E-06
	Acc	**0,971** ±0,008	0,963 ±0,007	0,963 ±0,009	0,931 ±0,011
Glass	***t(s)***	**4,416** ±0,049	4,503 ±0137	4,637 ±0,046	4,668 ±0,076
	L	1,245E+00 ±0,100	9,938E-02 ±0,065	2,783E-01 ±0,180	**1,147E-02** ±0,009
	Acc	0,401 ±0,110	0,704 ±0,053	0,676 ±0,079	**0,758** ±0,066
Banknote	***t(s)***	**4,610** ±0,164	4,662 ±0,062	4,845 ±0,062	4,871 ±0,141
	L	2,673E-01 ±0,084	4,591E-02 ±0,171	4,806E-03 ±0,004	**4,173E-07** ±7,90E-07
	Acc	0,962 ±0,032	0,966 ±0,115	0,998 ±0,009	**0,999** ±0,002
Ionosphere	***t(s)***	**4,580** ±0,097	4,636 ±0,087	4,781 ±0,084	4,731 ±0,137
	L	2,919E-01 ±0,089	2,744E-02 ±0,031	4,226E-02 ±0,036	**7,790E-06** ±1,10E-05
	Acc	0,892 ±0,047	**0,917** ±0,043	0,897 ±0,045	0,895 ±0,026
Sonar	***t(s)***	**4,556** ±0,066	4,588 ±0,047	4,770 ±0,055	4,731 ±0,137
	L	1,295E-01 ±0,193	8,252E-05 ±1,21E-04	9,603E-06 ±4,13E-06	**4,648E-14** ±1,71E-013
	Acc	0,998 ±0,002	**1,000** ±0,000	**1,000** ±0,000	**1,000** ±0,000
Wine	***t(s)***	**4,242** ±0,066	4,384 ±0,055	4,384 ±0,064	4,566 ±0,063
	L	3,409E-02 ±0,004	5,128E-06 1,44E-06	9,926E0-05 ±1,70E-05	**5,069E-06** ±5,32E-05
	Acc	0,996 ±0,010	**1,000** ±0,000	0,998 ±0,007	0,996 ±0,009

Na Tabela 7 pode-se ver que os resultados do método proposto para o treinamento local dos clientes, apresentou valores (média ± desvio padrão) consistentes, e que competem com os métodos que representam o estado-da-arte da literatura.

A Figura 13 compara os valores de tempo de treinamento da Tabela 7, com o PSO-SGD obtendo o melhor desempenho só no conjunto de dados Iris, para os restantes dos outros seis conjuntos de dados o melhor desempenho foi do AdaGrad em termos de tempo de execuções experimentais. Os valores do PSO-SGD não são muito diferentes dos outros métodos de otimização, apesar de seus cálculos envolverem não só a quantidade de parâmetros das redes neurais e número de iterações, mas também envolve a quantidade de partículas, que neste caso são 25 partículas, o que faz com que a quantidade de avaliações a serem realizadas no PSO-SGD supere em muito a quantidade de avaliações dos outros métodos. De modo geral, a Figura 13 demonstra resultados similares entre os algoritmos, no que diz respeito ao tempo de treinamento.

A Figura 14 compara os valores de erros obtidos da Tabela 7, mostrando um desempenho melhor do PSO-SGD em todos os sete conjuntos de dados. Na Figura 14, as colunas representando o PSO-SGD não são visíveis em alguns conjunto de dados, devido

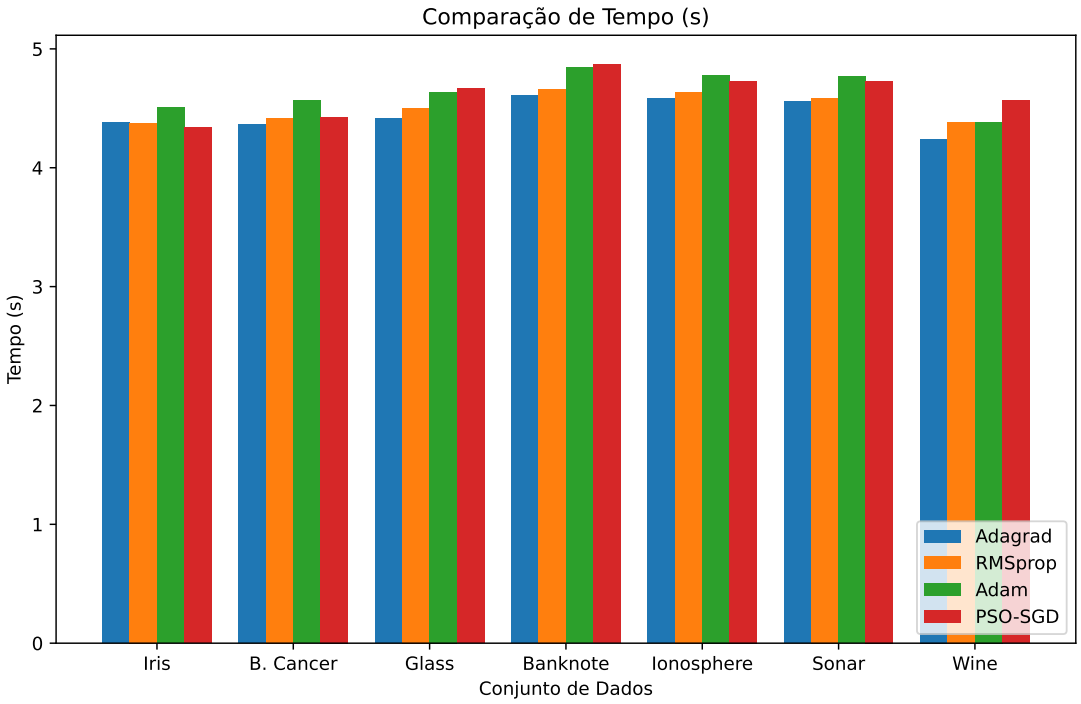

Figura 13 – Comparação do tempo de treinamento(s).

ao seu valor ser muito próximo ao zero e diferente em termos de escala dos erros dos outros métodos. Essa vantagem sobre os outros métodos, se deve a natureza híbrida do PSO-SGD, que consegue usar as vantagens tanto do PSO quanto do SGD, e assim, conseguir produzir altos desempenhos em termos dos erros de validação em comparação com os outros métodos utilizados. O Adam foi o algoritmo com o segundo melhor desempenho em termos do erro.

A Figura 15 compara os valores de acurácia obtidos na Tabela 7. PSO-SGD venceu em três conjuntos de dados *(Iris, Glass e Banknote)*, os resultados são similares no conjunto de dados *(Sonar)* e teve menor desempenho para outros métodos em três conjunto de dados *(B. Cancer, Ionosphere e Wine)*. Para *B. Cancer Adagrad* obteve maior valor de acurácia e para *Inosphere* e *Wine* o método de otimização que obteve maior valor de acurácia foi o *RMSprop*. Esses valores sugerem a efetividade do método PSO-SGD e uma alta competitividade em relação a outros otimizadores da literatura, isto é, com melhores desempenho em três conjuntos de dados. Em termos de acurácia os resultados dos algoritmos comparados são relativamente similares.

Para as comparações das aplicações de CNNs na fase 1, devido a complexidade das arquiteturas convolucionais e limites de recursos computacionais, o PSO-SGD foi comparado ao método Adam (que obteve o segundo melhor resultados em temos de erros na Tabela 7) nas quatro arquiteturas definidas na Tabela 6.

A Figura 16 compara o erro e a acurácia do PSO-SGD e Adam aplicado ao

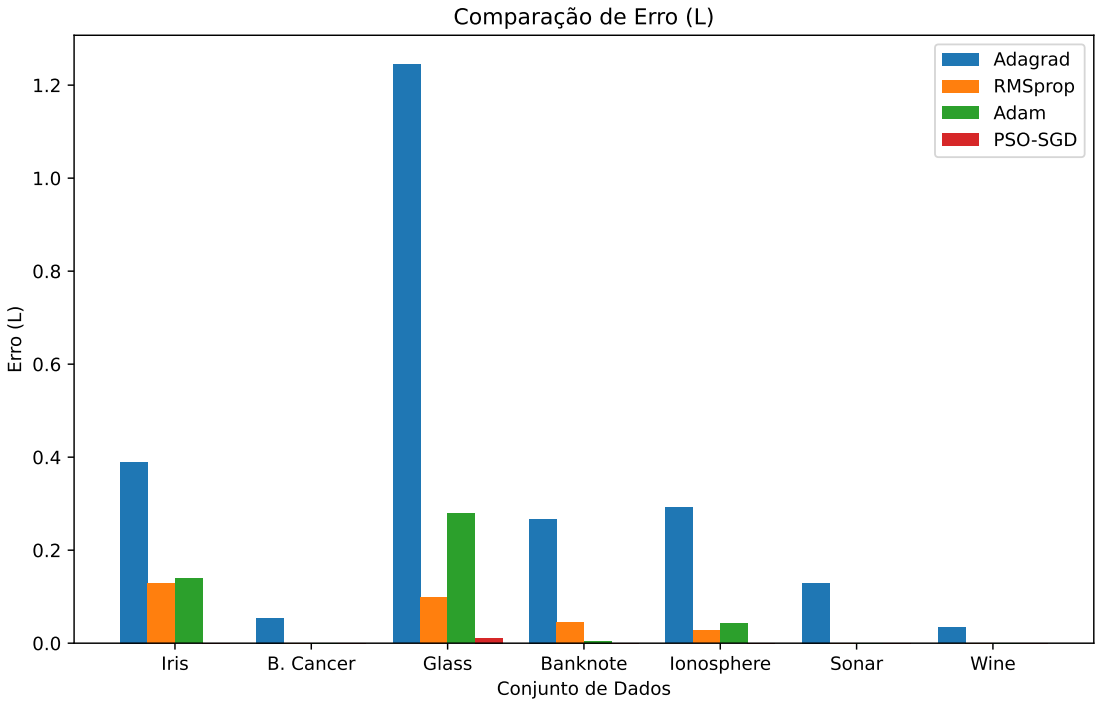

Figura 14 – Comparação dos erros de validação.

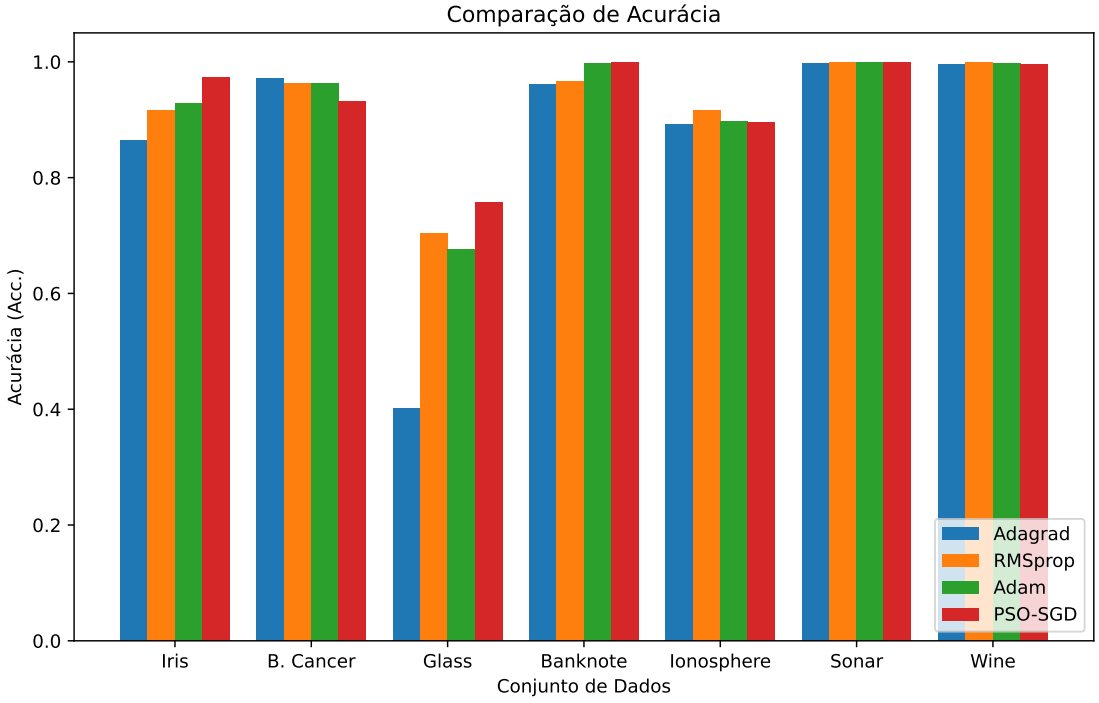

Figura 15 – Comparação da acurácia.

problema CIFAR-10 na arquitetura *SqueezeNet*, mostrando alto nível de competitividade e similaridade de comportamento em termos gerais, para os dois métodos de treinamento local. O PSO-SGD conseguiu um erro mínimo 0,6353 e enquanto que o mínimo de erro obtido por Adam nessa execução é 0,6444. A acurácia máxima conseguida para os dois métodos foram 78,89% para PSO-SGD e 78,28% para Adam. O PSO-SGD durante as rodadas de comunicação apresenta uma ligeira vantagem sobre o Adam nessa aplicação, entretanto, na última época, tanto o erro quanto a acurácia houve uma inversão onde o Adam obteve uma vantagem.

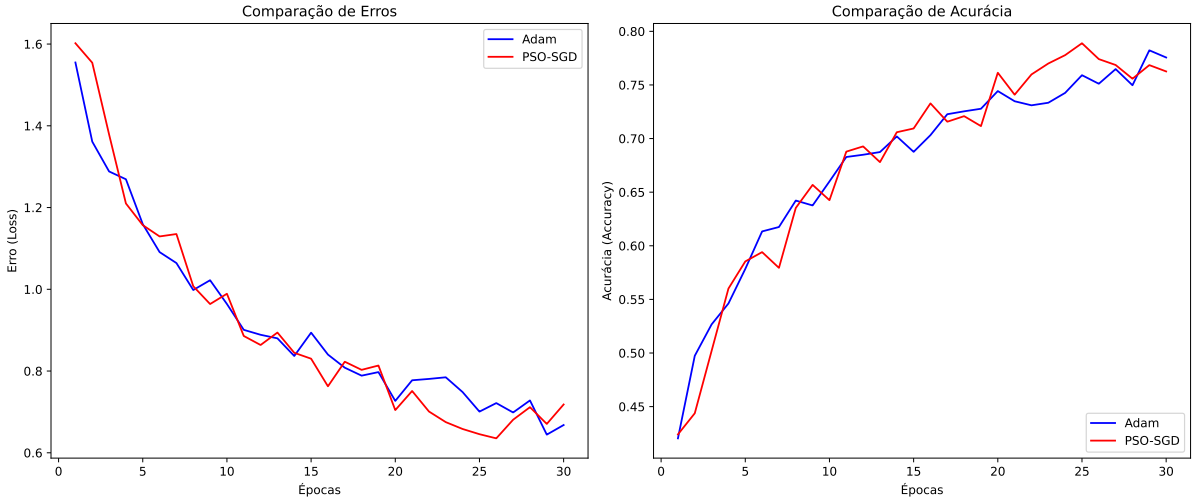

Figura 16 – Comparação SqueezeNet.

A Figura 17 faz as comparações no âmbito da arquitetura *MobileNetV2*, os resultados são relativamente diferentes aos valores da arquitetura anterior, entretanto, os mínimos e os máximos são próximos. Ou seja, o PSO-SGD conseguiu um erro mínimo 0,7161 e enquanto que o mínimo de erro conseguido por Adam nessa execução é 0,7397. A acurácia máxima conseguida para os dois métodos foram 78,44% para PSO-SGD e 78,06% para Adam. Nessa arquitetura o PSO-SGD e o Adam obtiveram resultados relativamente similares tanto para o erro, quanto para a acurácia.

A Figura 18 faz as comparações no âmbito da arquitetura *ResNet18*, o PSO-SGD conseguiu um erro mínimo 0,6777 e enquanto que o mínimo de erro conseguida por Adam nessa execução é 0,6706. A acurácia máxima conseguida para os dois métodos foram 81,08% para PSO-SGD e 80,66% para Adam. Novamente *ResNet18* PSO-SGD e Adam seguem um desempennho similar, apesar dos gráficos dessa figura mostrarem uma instabilidade (oscilações dos dois algoritmos), isto devido, à baixa taxa de aprendizado (0,0001), como descrito na Tabela 5, sendo esses valores superiores a *baseline* do CIFAR-10 para CNNs (HUANG et al., 2019).

Para a aplicação com a arquitetura *AlexNet*, os resultados da Figura 19 mostram que o PSO-SGD conseguiu um erro mínimo 0,4856 e enquanto que o mínimo de erro

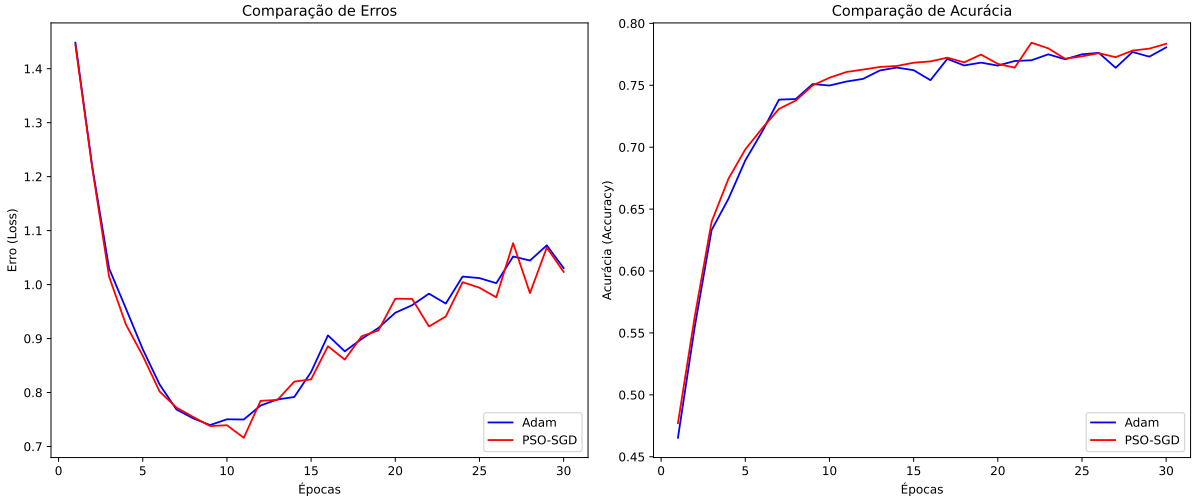

Figura 17 – Comparação *MobileNetV2*.

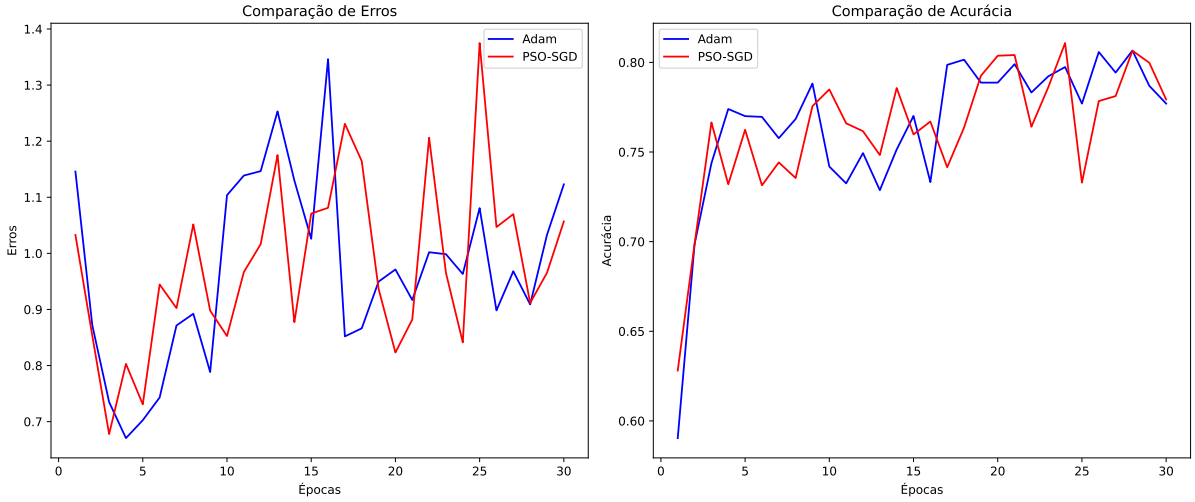

Figura 18 – Comparação ResNet18.

conseguida por Adam nessa execução é 0,5458. A acurácia máxima conseguida para os dois métodos foram 86,28% para PSO-SGD e 84,71% para Adam. O PSO-SGD obteve a vantagem relativas nessa arquitetura durante toda a execução experimental, com valores muito mais estáveis em termos de acurácia em comparação com o Adam.

Os resultados obtidos nesta fase experimental demonstram que o método proposto PSO-SGD para o treinamento local é viável e muito competitivo em relação a outros métodos de treinamento local, possuindo vantagens em diversos problemas de classificação. E também, em diversas arquiteturas convolucionais aplicadas ao problema CIFAR-10.

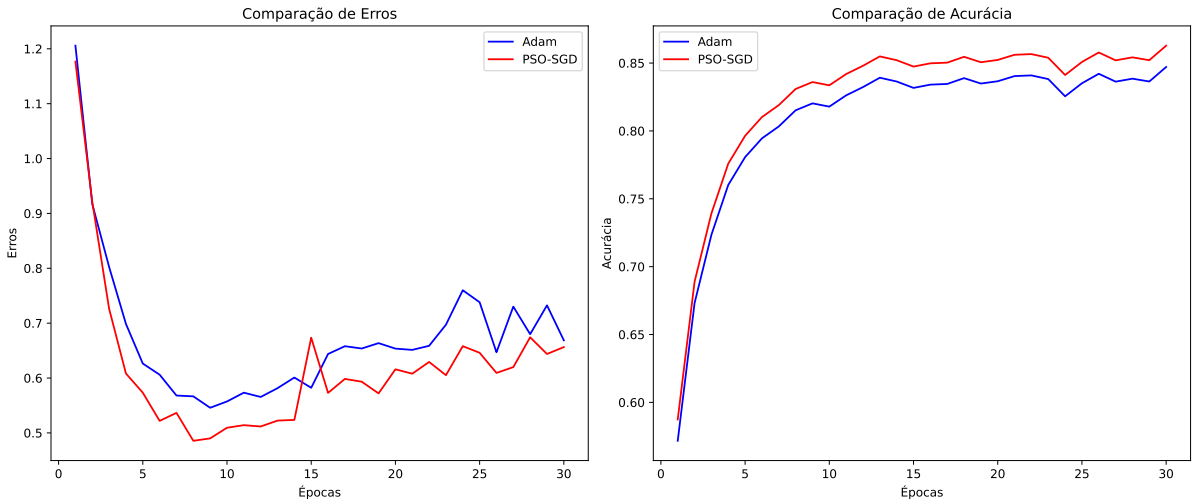

Figura 19 – Comparação AlexNet.

4.4.2 Fase Experimental 2

Os resultados obtidos nesta fase experimental aplicando o conjunto de dados de teste do CIFAR-10, correspondem a dois tipos de experimentos utilizando os algoritmos federados FedAvg, FedPSO e FLPSO-SGD: o primeiro tipo de experimento consiste na utilização do conjunto completo de dados CIFAR-10 com os modelos treinados do zero; e enquanto que o segundo tipo de experimento dessa fase, foi realizada com dados do CIFAR-10 particionados de acordo com os números dos clientes em conjuntos de amostras não IID e utilizando as arquiteturas CNNs com modelos correspondentes pré-treinados.

4.4.2.1 Treinamento do zero

Devido a custo computacional, em termos de tempo de treinamento, nestes experimentos utilizou-se três arquiteturas convolucionais, ou seja, os treinamentos foram realizados com *SqueezeNet, MobileNetV2 e ResNet18*.

A Figura 20 compara os algoritmos de agregação em uma arquitetura *SqueezeNet*, mostrando uma clara vantagem de FLPSO-SGD sobre os outros métodos, isto é, o FLPSO-SGD conseguiu o menor erro 0,5690 e a maior acurácia 81,94%; FedAvg conseguiu como menor erro 0,8466 e sua maior acurácia foi de 74,54%; e por fim FedPSO conseguiu o segundo melhor desempenho possuindo o menor erro 0,6234 e como maior acurácia 79,41%. No que diz respeito a redução de tamanho de parâmetros transferidos do cliente para o servidor, tendo em conta que os clientes por rodada são 10, e a quantidade de parâmetros da *SqueezeNet* é 727.626, então, FedAvg transfere dos clientes para o servidor por dez vezes mais parâmetros que o FedPSO e FLPSO-SGD, ou seja, FedAvg transfere para o servidor 27,8 MB por rodada enquanto que FedPSO e FLPSO-SGD transferem 2,78 MB

por rodada, com FLPSO-SGD possuindo um desempenho melhor.

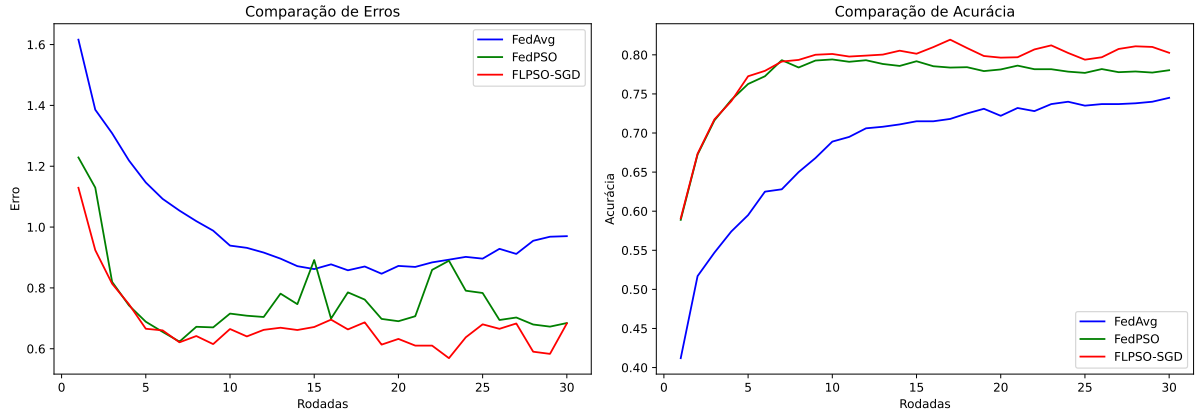

Figura 20 – Comparação dos algoritmos federados na arquitetura *SqueezeNet* treinada do zero.

Para *MobileNetV2*, ver Figura 21, os resultados de FLPSO-SGD são melhores em termos do erro e compete com FedPSO em termos de acurácia. O algoritmo FLPSO-SGD obteve o menor erro 0,7360 e a maior acurácia 80,93%; FedAvg com o menor erro 1,1602 e sua maior acurácia foi de 74,00%; e por fim FedPSO obteve o menor erro 0,8564 e a maior acurácia 80,86%. A CNN *MobileNetV2* utilizada possui 2.236.682 parâmetros como mostrado na Tabela 6, com isso, FedAvg transfere dos clientes para o servidor 85,0 MB por rodada enquanto que FedPSO e FLPSO-SGD transferem 8,50 MB por rodada.

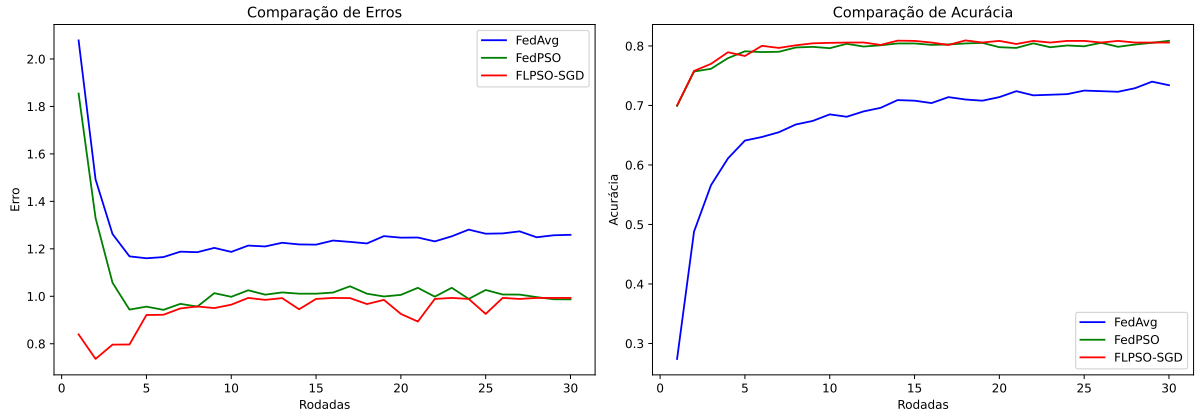

Figura 21 – Comparação dos algoritmos federados na arquitetura *MobileNetV2* treinada do zero.

A Figura 22 mostra os resultados de FLPSO-SGD com desempenho superior aos outros métodos. O algoritmo FLPSO-SGD obteve o menor erro 0,6169 e a maior acurácia 84,37%; FedPSO obteve o segundo melhor desempenho com o menor erro 0,6425 e a maior acurácia 83,05%; enquanto FedAvg com o pior desempenho obteve 1,1462 como o menor erro e sua maior acurácia foi de 73,00%. Para a CNN *ResNet18*, FLPSO-SGD e FedPSO

transmitem dos clientes para o servidor 42,7 MB por rodada e enquanto que FedAvg transmite 427 MB por rodada.

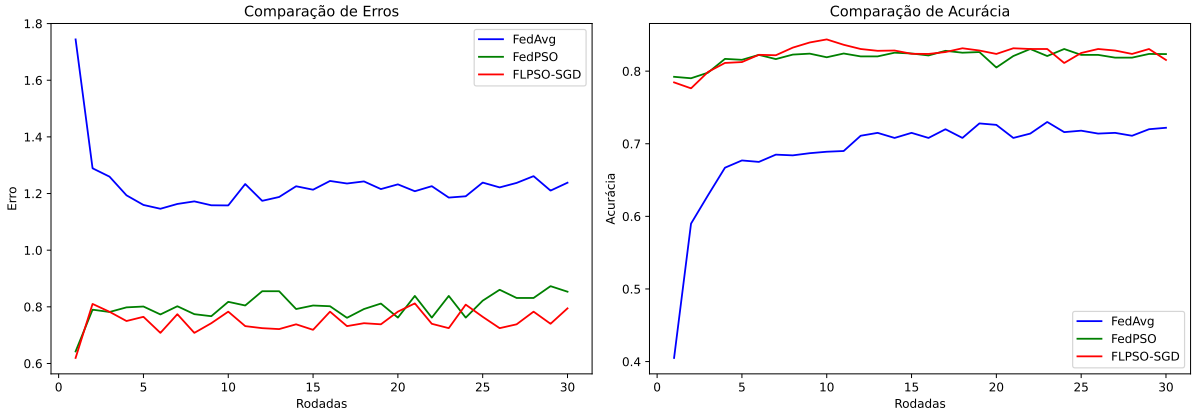

Figura 22 – Comparação dos algoritmos federados na arquitetura *ResNet18* treinada do zero.

Os resultados obtidos nesses primeiros experimentos da segunda fase, demonstraram-se coerentes com os resultados apresentados em outros trabalhos relacionados (PARK; SUH; LEE, 2021; VICTOR et al., 2023; BAKIR; SAMROUTH; SAMROUTH, 2023), com FLPSO-SGD obtendo consideráveis vantagens em termos de acurácias e erros devido ao treinamento local dos clientes mais eficientes em diferentes CNNs utilizadas.

4.4.2.2 Treinamento com modelos pré-treinados

Os resultados desta subseção correspondem ao segundo tipo de experimentos realizados na segunda fase experimental do presente trabalho. As quatro arquiteturas CNNs descritas na Tabela 6 foram utilizadas para treinar os conjuntos de dados com a presença de heterogeneidade estatística, ou seja, o conjunto de dados CIFAR-10 divido por número de clientes em distribuição não homogênea.

A Figura 23 compara os resultados obtidos pelos três métodos federados com dados desbalanceados na arquitetura *SqueezeNet* com os parâmetros pré-treinados. O FLPSO-SGD obteve melhores resultados tanto em termos dos erros, quanto em termos de acurácia, conseguindo um erro mínimo de 0,3318 e a acurácia máxima de 91,8%. Por outro lado, o FedPSO (erro mínimo 0,4217 e acurácia máxima 86,81%) e FedAvg (erro mínimo 0,4157 e acurácia máxima 86,52%) obtiveram os resultados de erros e acurácias piores e convergências mais lentas que FLPSO-SGD.

Os resultados relativos à arquitetura *MobileNetV2* mostram uma instabilidade maior e convergência mais lenta nos métodos FedAvg e FedPSO comparados ao FLPSO-SGD, como ilustra os gráficos da Figura 24. Isto se deve, em grande parte à incapacidade dos dois primeiros métodos de lidarem com dados desbalanceados. Com a utilização de uma

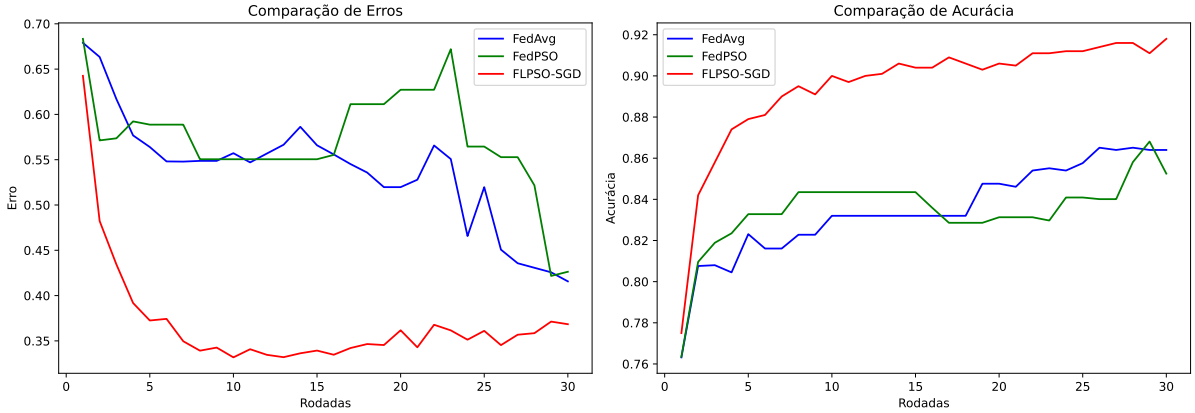

Figura 23 – Comparação dos algoritmos federados na arquitetura *SqueezeNet* com modelos pré-treinados.

arquitetura com modelos pré-treinados FedPSO (erro mínimo 0,3374 e acurácia máxima de 90,1%) e FedAvg (erro mínimo 0,2961 e acurácia máxima de 91,0%) conseguem atenuar os impactos de dados não homogêneos e obtiveram resultados acima de *baseline* estabelecido pelo trabalho de Darapaneni, Krishnamurthy e Paduri (2020). Entretanto, o algoritmo FLPSO-SGD demonstrou melhor desempenho, onde obteve um erro mínimo de 0,2537 e uma acurácia máxima de 94,6%, e também convergência mais rápida e estabilidade melhor.

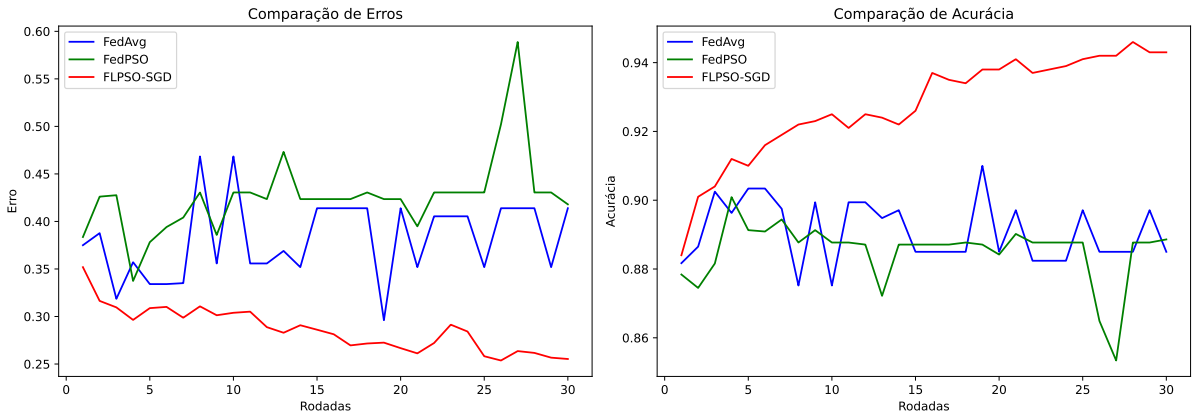

Figura 24 – Comparação dos algoritmos federados na arquitetura *MobileNetV2* com modelos pré-treinados.

Na arquitetura *ResNet18*, Figura 25, o desempenho do FLPSO-SGD em termos de erro e acurácia continuam superiores ao desempenho do FedPSO e FedAvg. Isto é, o FLPSO-SGD possui um erro de 0,1246 menor que FedPSO, 0,115 menor que FedAvg, a acurácia 4,7 pp maior que FedPSO e 4,5 pp maior que FedAvg. A convergência e a estabilidade do FLPSO-SGD são melhores que os referidos métodos, como ilustra a figura em questão.

A última arquitetura com parâmetros pré-treinados testada, foi *AlexNet*, cuja a

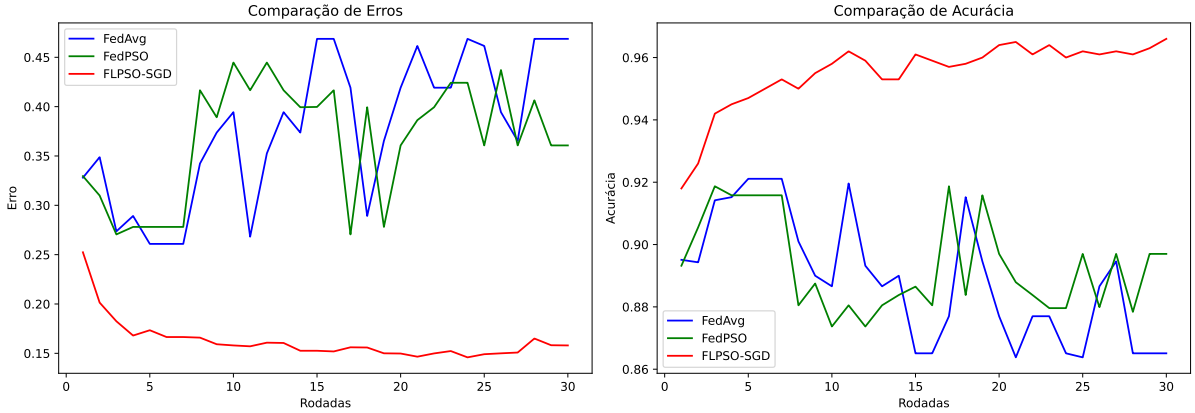

Figura 25 – Comparação dos algoritmos federados na arquitetura *ResNet18* com modelos pré-treinados.

comparação do aprendizado das três metodologias federadas estão ilustradas na Figura 26. Como nas arquiteturas anteriores, apesar do modelo com parâmetros previamente treinados conseguir atenuar o impacto de dados desbalanceados, todos os métodos comparados obtiveram resultados coerentes e superiores à linha de base estabelecido por Darapaneni, Krishnamurthy e Paduri (2020) para *AlexNet*. Entretanto, FLPSO-SGD (erro mínimo 0,2789 e acurácia máxima 92,1%) obteve vantagens sobre FedPSO (erro mínimo 0,5729 e acurácia máxima 85,5%) e FedAvg (erro mínimo 0,4994 e acurácia máxima 86,02%).

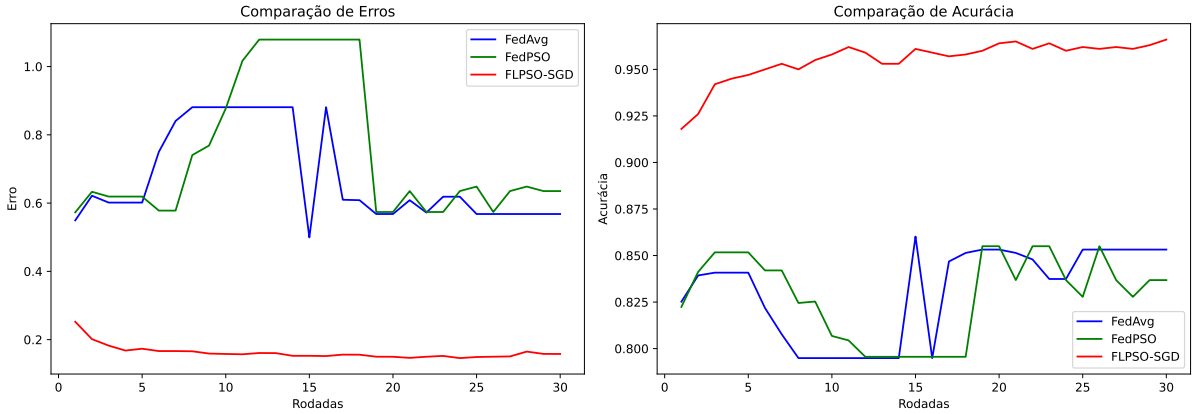

Figura 26 – Comparação dos algoritmos federados na arquitetura *AlexNet* com modelos pré-treinados.

A comparação dos três métodos supra mencionados, com dados desbalanceados em diferentes arquiteturas convolucionais pré-treinadas, demonstrou que o FLPSO-SGD consegue ter melhor desempenho comparado aos outros métodos ao lidar com dados não IID. Já que o método do treinamento local dos clientes com PSO-SGD, consegue escapar de mínimos locais mais facilmente e mantendo um exploração mais rápida, enquanto que o modelo global *gbest* consegue atenuar ligeiramente os impactos dos dados não homogêneos.

Estes dois aspectos trabalham em conjunto para mitigar em grande medida a instabilidade e a convergência mais lenta que é verificada no FedPSO e FedAvg.

5 CONCLUSÃO

As abordagens de FL emergem como uma solução robusta para treinamentos colaborativos e com dados distribuídos, assegurando a privacidade e segurança dos dados dos dispositivos. Este estudo apresentou o FLPSO-SGD como uma nova metodologia para o treinamento federado, utilizando o método híbrido PSO-SGD como algoritmo local nos clientes. Os resultados destacam o desempenho promissor do algoritmo PSO-SGD, evidenciado por seu tempo de treinamento eficiente, baixo erro de validação e alta acurácia. Além disso, a metodologia federada FLPSO-SGD demonstrou acurácias superiores no treinamento global em comparação com as técnicas FedAvg e FedPSO. Consequentemente, esses achados sugerem que a nova abordagem federada não melhora apenas o custo de comunicação, mas também eleva o desempenho global do sistema, consolidando-se como uma alternativa viável e eficaz para o treinamento distribuído em ambientes federados.

Para o treinamento local dos clientes, o PSO-SGD obteve resultados competitivo ou com uma vantagem relativa de desempenho comparado a outras metodologias atuais da literatura em diferentes problemas clássicos de ML utilizando uma MLP simples. E para o problema de CIFAR-10 utilizando diversas arquiteturas convolucionais, PSO-SGD apresentou valores de desempenho similares ao do algoritmo de otimização Adam, o que viabiliza o PSO-SGD como uma opção para o treinamento local do clientes em ambientes de aprendizado federado. Uma vantagem adicional que se apresentou durante a pesquisa é a natureza de componentes de PSO possuir propriedades que conseguem regularizar as influências dos modelos locais e globais, evitando assim, mudanças abruptas que possam degradar o modelo global fornecidos pelo servidor.

O FLPSO-SGD proposto no presente trabalho se mostrou uma boa opção para o treinamento federado, pois obteve melhores resultados nos experimentos, comparado a FedAvg e FedPSO, em diferentes arquiteturas convolucionais tanto para conjunto CIFAR-10 IDD quanto para não IID. Os resultados para treinamento com dados completos do CIFAR-10 demonstrou uma clara vantagem em termos de erro e acurácia do FLPSO-SGD comparado a outros dois métodos.

Para o treinamento com dados particionados em conjuntos de amostras não IID, os resultados do FLPSO-SGD possuem vantagens claras e expressivas, demonstrando um desempenho significativamente melhor ao lidar com os dados desbalanceados comparado ao FedPSO e ao FedAvg, seja em termos de erro, acurácia, quanto em termos, de convergência e da estabilidade do modelo o FLPSO-SGD obteve desempenho superior.

FLPSO-SGD e FedPSO reduziram a quantidade de parâmetros transferidos do cliente para o servidor em dez vezes comparado a FedAvg. Entretanto, FLPSO-SGD

possui desempenho melhor que FedPSO nas CNNs experimentadas. Ou seja, o método de treinamento em FL proposto no presente trabalho possui uma eficiência melhor que FedAvg e FedPSO, e ainda possui uma alta redução de custo de comunicação igual ao FedPSO.

As principais dificuldades encontradas durante a pesquisa foram os recursos computacionais limitados, o que impossibilitou maiores experimentos: isto é, experimentos com conjunto de dados maiores e arquiteturas CNNs muito mais complexas e profundas. A metodologia proposta apesar de conseguir atenuar ataques ao modelo, a sua efetividade contra outros tipos de possíveis ataques pesados dos "clientes maliciosos"ainda precisa ser melhor avaliado experimentalmente.

As perspectivas para trabalhos futuros são o desenvolvimento da versão do FLPSO-SGD descentralizada com privacidade diferencial, baseada em *Local Best PSO*, isto é, utilizando as redes ponto-a-ponto de tecnologia blockchain, dispensando assim a utilização do servidor central. Esta abordagem poderia suprimir em grande medida ataques contra privacidade dos clientes, ataques contra o modelo global e ataques contra os dados dos clientes.

REFERÊNCIAS

ALPAYDIN, E. Introduction to machine learning. [S.l.]: MIT press, 2020.

AUGENSTEIN, S. et al. Generative models for effective ml on private, decentralized datasets. arXiv preprint arXiv:1911.06679, 2019.

AZIZ, R. et al. Exploring homomorphic encryption and differential privacy techniques towards secure federated learning paradigm. Future internet, MDPI, v. 15, n. 9, p. 310, 2023.

BAKIR, N.; SAMROUTH, A.; SAMROUTH, K. Pso-ga-based federated learning for predicting energy consumption in smart buildings. In: IEEE. 2023 International Conference on Microelectronics (ICM). [S.l.], 2023. p. 234–238.

BEJNORDI, B. E. et al. Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer. Jama, American Medical Association, v. 318, n. 22, p. 2199–2210, 2017.

BENVIDI, A. et al. Spectrophotometric determination of synthetic colorants using pso–ga-ann. Food chemistry, Elsevier, v. 220, p. 377–384, 2017.

BOJARSKI, M. et al. End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316, 2016.

CHOI, D. et al. On empirical comparisons of optimizers for deep learning. arXiv preprint arXiv:1910.05446, 2019.

CHUNG, J. et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555, 2014.

CLERC, M. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In: IEEE. Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406). [S.l.], 1999. v. 3, p. 1951–1957.

CLERC, M.; KENNEDY, J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE transactions on Evolutionary Computation, IEEE, v. 6, n. 1, p. 58–73, 2002.

CONRADIE, A. E.; MIIKKULAINEN, R.; ALDRICH, C. Intelligent process control utilising symbiotic memetic neuro-evolution. In: IEEE. Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600). [S.l.], 2002. v. 1, p. 623–628.

CORNE, D. W.; REYNOLDS, A. P.; BONABEAU, E. Swarm intelligence. Handbook of natural computing, v. 2017, n. 6, p. 1599–1622, 2012.

DAI, S.; MENG, F. Addressing modern and practical challenges in machine learning: A survey of online federated and transfer learning. Applied Intelligence, Springer, v. 53, n. 9, p. 11045–11072, 2023.

DARAPANENI, N.; KRISHNAMURTHY, B.; PADURI, A. R. Convolution neural networks: A comparative study for image classification. In: IEEE. 2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS). [S.l.], 2020. p. 327–332.

DARWISH, A.; HASSANIEN, A. E.; DAS, S. A survey of swarm and evolutionary computing approaches for deep learning. Artificial intelligence review, Springer, v. 53, p. 1767–1812, 2020.

DAS, S.; ABRAHAM, A.; KONAR, A. Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives. Advances of computational intelligence in industrial systems, Springer, p. 1–38, 2008.

DEAN, J. et al. Large scale distributed deep networks. Advances in neural information processing systems, v. 25, 2012.

DEVLIN, J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. Journal of machine learning research, v. 12, n. 7, 2011.

ESTEVA, A. et al. Dermatologist-level classification of skin cancer with deep neural networks. nature, Nature Publishing Group, v. 542, n. 7639, p. 115–118, 2017.

FORREST, S. Genetic algorithms. ACM computing surveys (CSUR), ACM New York, NY, USA, v. 28, n. 1, p. 77–80, 1996.

GHAMISI, P.; BENEDIKTSSON, J. A. Feature selection based on hybridization of genetic algorithm and particle swarm optimization. IEEE Geoscience and remote sensing letters, IEEE, v. 12, n. 2, p. 309–313, 2014.

GOODFELLOW, I. et al. Generative adversarial nets. Advances in neural information processing systems, v. 27, 2014.

GU, S. et al. Continuous deep q-learning with model-based acceleration. In: PMLR. International conference on machine learning. [S.l.], 2016. p. 2829–2838.

HAARNOJA, T. et al. Composable deep reinforcement learning for robotic manipulation. In: IEEE. 2018 IEEE international conference on robotics and automation (ICRA). [S.l.], 2018. p. 6244–6251.

HAMER, J.; MOHRI, M.; SURESH, A. T. Fedboost: A communication-efficient algorithm for federated learning. In: PMLR. International Conference on Machine Learning. [S.l.], 2020. p. 3973–3983.

HAO, Z.-F.; GUO, G.-H.; HUANG, H. A particle swarm optimization algorithm with differential evolution. In: IEEE. 2007 international conference on machine learning and cybernetics. [S.l.], 2007. v. 2, p. 1031–1035.

HE, H.; LIN, J. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In: Proceedings of the 2016 conference of the north American chapter of the Association for Computational Linguistics: human language technologies. [S.l.: s.n.], 2016. p. 937–948.

HE, K. et al. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2016. p. 770–778.

HENDTLASS, T. A combined swarm differential evolution algorithm for optimization problems. In: SPRINGER. Engineering of Intelligent Systems: 14th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE 2001 Budapest, Hungary, June 4–7, 2001 Proceedings 14. [S.l.], 2001. p. 11–18.

HINTON, G.; SRIVASTAVA, N.; SWERSKY, K. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. Cited on, v. 14, n. 8, p. 2, 2012.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. Neural computation, MIT press, v. 9, n. 8, p. 1735–1780, 1997.

HUANG, C. et al. Promoting collaborations in cross-silo federated learning: Challenges and opportunities. IEEE Communications Magazine, IEEE, 2023.

HUANG, C.-L. et al. Optimization of a convolutional neural network using a hybrid algorithm. In: IEEE. 2019 International Joint Conference on Neural Networks (IJCNN). [S.l.], 2019. p. 1–8.

HUANG, H. et al. The back analysis of mechanics parameters based on depso algorithm and parallel fem. In: IEEE. 2009 International Conference on Computational Intelligence and Natural Computing. [S.l.], 2009. v. 1, p. 81–84.

IANDOLA, F. N. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. arXiv preprint arXiv:1602.07360, 2016.

ISSA, W. et al. Blockchain-based federated learning for securing internet of things: A comprehensive survey. ACM Computing Surveys, ACM New York, NY, v. 55, n. 9, p. 1–43, 2023.

JEBREEL, N. M.; DOMINGO-FERRER, J. Fl-defender: Combating targeted attacks in federated learning. Knowledge-Based Systems, Elsevier, v. 260, p. 110178, 2023.

JUANG, C.-F. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), IEEE, v. 34, n. 2, p. 997–1006, 2004.

KARIMIREDDY, S. P. et al. Scaffold: Stochastic controlled averaging for federated learning. In: PMLR. International conference on machine learning. [S.l.], 2020. p. 5132–5143.

KAYA, U.; YILMAZ, A.; AŞAR, S. Sepsis prediction by using a hybrid metaheuristic algorithm: A novel approach for optimizing deep neural networks. Diagnostics, MDPI, v. 13, n. 12, p. 2023, 2023.

KENNEDY, J. Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In: IEEE. Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406). [S.l.], 1999. v. 3, p. 1931–1938.

KENNEDY, J.; EBERHART, R. Particle swarm optimization (pso). In: Proc. IEEE international conference on neural networks, Perth, Australia. [S.l.: s.n.], 1995. v. 4, n. 1, p. 1942–1948.

KENNEDY, J.; MENDES, R. Population structure and particle swarm performance. In: IEEE. Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600). [S.l.], 2002. v. 2, p. 1671–1676.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

KINGMAN, D.; BA, J. Adam: A method for stochastic optimization. conference paper. In: 3rd International Conference for Learning Representations. [S.l.: s.n.], 2015. v. 2015, n. 2.

KONEČNỲ, J. et al. Federated optimization: Distributed machine learning for on-device intelligence. arXiv preprint arXiv:1610.02527, 2016.

KRINK, T.; LØVBJERG, M. The lifecycle model: combining particle swarm optimisation, genetic algorithms and hillclimbers. In: SPRINGER. International Conference on Parallel Problem Solving from Nature. [S.l.], 2002. p. 621–630.

KRIZHEVSKY, A.; HINTON, G. et al. Learning multiple layers of features from tiny images. Toronto, ON, Canada, 2009.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, v. 25, 2012.

LI, C. et al. Optimization of a heliostat field layout using hybrid pso-ga algorithm. Applied Thermal Engineering, Elsevier, v. 128, p. 33–41, 2018.

LI, Q. et al. A survey on federated learning systems: Vision, hype and reality for data privacy and protection. IEEE Transactions on Knowledge and Data Engineering, IEEE, v. 35, n. 4, p. 3347–3366, 2021.

LI, T. et al. Federated learning: Challenges, methods, and future directions. IEEE signal processing magazine, IEEE, v. 37, n. 3, p. 50–60, 2020.

LI, T. et al. Federated optimization in heterogeneous networks. Proceedings of Machine learning and systems, v. 2, p. 429–450, 2020.

LI, W.-T.; XU, L.; SHI, X.-W. A hybrid of genetic algorithm and particle swarm optimization for antenna design. PIERS online, PIERS Online, v. 4, n. 1, p. 56–60, 2008.

LIU, P.; XU, X.; WANG, W. Threats, attacks and defenses to federated learning: issues, taxonomy and perspectives. Cybersecurity, Springer, v. 5, n. 1, p. 4, 2022.

LIU, Q.-j. et al. Ensemble feature selection method based on neighborhood information and pso algorithm. ACTA ELECTONICA SINICA, v. 44, n. 4, p. 995, 2016.

LIU, Y. et al. Fedvision: An online visual object detection platform powered by federated learning. In: Proceedings of the AAAI conference on artificial intelligence. [S.l.: s.n.], 2020. v. 34, n. 08, p. 13172–13179.

LUITEL, B.; VENAYAGAMOORTHY, G. K. Differential evolution particle swarm optimization for digital filter design. In: IEEE. 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence). [S.l.], 2008. p. 3954–3961.

LV, Z. et al. Awfc: Preventing label flipping attacks towards federated learning for intelligent iot. The Computer Journal, Oxford University Press, v. 65, n. 11, p. 2849–2859, 2022.

MANZOOR, H. U. et al. Enhanced adversarial attack resilience in energy networks through energy and privacy aware federated learning. Authorea Preprints, Authorea, 2024.

MANZOOR, H. U. et al. Centralised vs. decentralised federated load forecasting: Who holds the key to adversarial attack robustness? Authorea Preprints, Authorea, 2024.

MANZOOR, H. U. et al. Defending federated learning from backdoor attacks: Anomaly-aware fedavg with layer-based aggregation. In: IEEE. 2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC). [S.l.], 2023. p. 1–6.

MANZOOR, H. U. et al. A survey of security strategies in federated learning: Defending models, data, and privacy. Future Internet, MDPI, v. 16, n. 10, p. 374, 2024.

MCMAHAN, B. et al. Communication-efficient learning of deep networks from decentralized data. In: PMLR. Artificial intelligence and statistics. [S.l.], 2017. p. 1273–1282.

MEHMOOD, F.; AHMAD, S.; WHANGBO, T. K. An efficient optimization technique for training deep neural networks. Mathematics, MDPI, v. 11, n. 6, p. 1360, 2023.

MENDES, R.; KENNEDY, J.; NEVES, J. Watch thy neighbor or how the swarm can learn from its environment. In: IEEE. Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706). [S.l.], 2003. p. 88–94.

MOHAMMADI, S. et al. Balancing privacy and performance in federated learning: A systematic literature review on methods and metrics. Journal of Parallel and Distributed Computing, Elsevier, p. 104918, 2024.

MOSHAWRAB, M. et al. Reviewing federated learning aggregation algorithms; strategies, contributions, limitations and future perspectives. Electronics, MDPI, v. 12, n. 10, p. 2287, 2023.

MOSHAWRAB, M. et al. Reviewing federated learning aggregation algorithms; strategies, contributions, limitations and future perspectives. Electronics, MDPI, v. 12, n. 10, p. 2287, 2023.

MOUSSA, R.; AZAR, D. A pso-ga approach targeting fault-prone software modules. Journal of Systems and Software, Elsevier, v. 132, p. 41–49, 2017.

MÜLLER, K.; BODENDORF, F. Cross-silo federated learning in enterprise networks with cooperative and competing actors. In: The Human Side of Service Engineering, AHFE 2023 International Conference, San Francisco, USA. [S.l.: s.n.], 2023.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10). [S.l.: s.n.], 2010. p. 807–814.

NEWTON, D.; YOUSEFIAN, F.; PASUPATHY, R. Stochastic gradient descent: Recent trends. Recent advances in optimization and modeling of contemporary problems, INFORMS, p. 193–220, 2018.

NIK, A. A.; NEJAD, F. M.; ZAKERI, H. Hybrid pso and ga approach for optimizing surveyed asphalt pavement inspection units in massive network. Automation in Construction, Elsevier, v. 71, p. 325–345, 2016.

OJHA, V. K.; ABRAHAM, A.; SNÁŠEL, V. Metaheuristic design of feedforward neural networks: A review of two decades of research. Engineering Applications of Artificial Intelligence, Elsevier, v. 60, p. 97–116, 2017.

PARK, S.; SUH, Y.; LEE, J. Fedpso: Federated learning using particle swarm optimization to reduce communication costs. Sensors, MDPI, v. 21, n. 2, p. 600, 2021.

PARSOPOULOS, K. E.; VRAHATIS, M. N. Unified particle swarm optimization in dynamic environments. In: SPRINGER. Workshops on Applications of Evolutionary Computation. [S.l.], 2005. p. 590–599.

PHONG, N. H.; SANTOS, A.; RIBEIRO, B. Pso-convolutional neural networks with heterogeneous learning rate. IEEE Access, IEEE, v. 10, p. 89970–89988, 2022.

POLYAK, B. T. Gradient methods for solving equations and inequalities. USSR Computational Mathematics and Mathematical Physics, Elsevier, v. 4, n. 6, p. 17–32, 1964.

PREMALATHA, K.; NATARAJAN, A. Discrete pso with ga operators for document clustering. International Journal of Recent Trends in Engineering, Citeseer, v. 1, n. 1, p. 20, 2009.

RATNAWEERA, A. Particle swarm optimization with self-adaptive acceleration coefficients. In: Proc. Int'l Conf. Fuzzy Syst. & Knowledge Discovery (FSKD 2002), Singapore, Nov. [S.l.: s.n.], 2002. v. 1, p. 264–268.

REDDI, S. et al. Adaptive federated optimization. arXiv preprint arXiv:2003.00295, 2020.

ROBBINS, H.; MONRO, S. A stochastic approximation method. The annals of mathematical statistics, JSTOR, p. 400–407, 1951.

ROBINSON, J.; SINTON, S.; RAHMAT-SAMII, Y. Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna. In: IEEE. IEEE Antennas and Propagation Society International Symposium (IEEE Cat. No. 02CH37313). [S.l.], 2002. v. 1, p. 314–317.

SANDLER, M. et al. Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2018. p. 4510–4520.

SENGUPTA, S.; BASAK, S.; PETERS, R. A. Particle swarm optimization: A survey of historical and recent developments with hybridization perspectives. Machine Learning and Knowledge Extraction, MDPI, v. 1, n. 1, p. 157–191, 2018.

SHAMI, T. M. et al. Particle swarm optimization: A comprehensive survey. Ieee Access, IEEE, v. 10, p. 10031–10061, 2022.

SHI, X. et al. An improved ga and a novel pso-ga-based hybrid algorithm. Information Processing Letters, Elsevier, v. 93, n. 5, p. 255–261, 2005.

SHI, Y.; EBERHART, R. A modified particle swarm optimizer. In: IEEE. 1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360). [S.l.], 1998. p. 69–73.

STORN, R. Differrential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, International Computer Science Institute, v. 11, 1995.

STORN, R.; PRICE, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization, Springer, v. 11, p. 341–359, 1997.

SUGANTHAN, P. N. Particle swarm optimiser with neighbourhood operator. In: IEEE. Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406). [S.l.], 1999. v. 3, p. 1958–1962.

SZEGEDY, C. et al. Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2016. p. 2818–2826.

TALBI, H.; BATOUCHE, M. Hybrid particle swarm with differential evolution for multimodal image registration. In: IEEE. 2004 IEEE International Conference on Industrial Technology, 2004. IEEE ICIT'04. [S.l.], 2004. v. 3, p. 1567–1572.

TAN, M.; LE, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In: PMLR. International conference on machine learning. [S.l.], 2019. p. 6105–6114.

TIELEMAN, T.; HINTON, G. Lecture 6.5-rmsprop, coursera: Neural networks for machine learning. University of Toronto, Technical Report, v. 6, 2012.

VAISAKH, K.; SRIDHAR, M.; MURTHY, K. L. Differential evolution particle swarm optimization algorithm for reduction of network loss and voltage instability. In: IEEE. 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC). [S.l.], 2009. p. 391–396.

VALDEZ, F.; MELIN, P.; CASTILLO, O. Evolutionary method combining particle swarm optimization and genetic algorithms using fuzzy logic for decision making. In: IEEE. 2009 IEEE International Conference on Fuzzy Systems. [S.l.], 2009. p. 2114–2119.

VARNO, F. et al. Adabest: Minimizing client drift in federated learning via adaptive bias estimation. In: SPRINGER. European Conference on Computer Vision. [S.l.], 2022. p. 710–726.

VICTOR, N. et al. Fl-pso: A federated learning approach with particle swarm optimization for brain stroke prediction. In: IEEE. 2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing Workshops (CCGridW). [S.l.], 2023. p. 33–38.

WAINAKH, A. et al. Federated learning attacks revisited: A critical discussion of gaps, assumptions, and evaluation setups. Sensors, MDPI, v. 23, n. 1, p. 31, 2022.

WANG, J. et al. Tackling the objective inconsistency problem in heterogeneous federated optimization. Advances in neural information processing systems, v. 33, p. 7611–7623, 2020.

WANG, T. et al. Applications of federated learning in mobile health: scoping review. Journal of Medical Internet Research, JMIR Publications Toronto, Canada, v. 25, p. e43006, 2023.

WEI, K. et al. Federated learning with differential privacy: Algorithms and performance analysis. IEEE transactions on information forensics and security, IEEE, v. 15, p. 3454–3469, 2020.

XU, Y.; PI, D. A reinforcement learning-based communication topology in particle swarm optimization. Neural Computing and Applications, Springer, v. 32, p. 10007–10032, 2020.

YADAV, R. K. et al. Pso-ga based hybrid with adam optimization for ann training with application in medical diagnosis. Cognitive Systems Research, Elsevier, v. 64, p. 191–199, 2020.

YANG, B.; CHEN, Y.; ZHAO, Z. A hybrid evolutionary algorithm by combination of pso and ga for unconstrained and constrained optimization problems. In: IEEE. 2007 IEEE International Conference on Control and Automation. [S.l.], 2007. p. 166–170.

YANG, H. et al. Gradient leakage attacks in federated learning: Research frontiers, taxonomy and future directions. IEEE Network, IEEE, 2023.

YANG, Q. et al. Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST), ACM New York, NY, USA, v. 10, n. 2, p. 1–19, 2019.

YU, S.; WEI, Y.-M.; WANG, K. A pso–ga optimal model to estimate primary energy demand of china. Energy Policy, Elsevier, v. 42, p. 329–340, 2012.

YUAN, L. et al. Decentralized federated learning: A survey and perspective. IEEE Internet of Things Journal, IEEE, 2024.

YUAN, X. et al. Beyond class-level privacy leakage: Breaking record-level privacy in federated learning. IEEE Internet of Things Journal, IEEE, v. 9, n. 4, p. 2555–2565, 2021.

ZAVALA, A. M.; AGUIRRE, A. H.; DIHARCE, E. V. Robust pso-based constrained optimization by perturbing the particle's memory. Swarm Intelligence. Focus on Ant and Particle Swarm Optimization, I-Tech Education and Publising, Croatia, p. 57–76, 2007.

ZHANG, W.; LIU, Y.; CLERC, M. An adaptive pso algorithm for reactive power optimization. IET, 2003.

ZHANG, W.-J.; XIE, X.-F. Depso: hybrid particle swarm with differential evolution operator. In: IEEE. SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme-System Security and Assurance (Cat. No. 03CH37483). [S.l.], 2003. v. 4, p. 3816–3821.

ZHAO, Y. et al. Federated learning with non-iid data. arXiv preprint arXiv:1806.00582, 2018.