UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE TELEINFORMÁTICA

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

GABRIELA SILVA DE CARVALHO PONCIANO

APRENDIZADO FEDERADO PARA CLASSIFICAÇÃO DA PRAGA DO MILHO EM CAMPO DE AGRICULTURA

FORTALEZA

2025

GABRIELA SILVA DE CARVALHO PONCIANO

APRENDIZADO FEDERADO PARA CLASSIFICAÇÃO DA PRAGA DO MILHO EM CAMPO DE AGRICULTURA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Orientadora: Profa. Dra. Atslands Rego da Rocha

FORTALEZA

2025

GABRIELA SILVA DE CARVALHO PONCIANO

APRENDIZADO FEDERADO PARA CLASSIFICAÇÃO DA PRAGA DO MILHO EM CAMPO DE AGRICULTURA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Aprovada em:

BANCA EXAMINADORA

Profa. Dra. Atslands Rego da Rocha (Orientadora)
Universidade Federal do Ceará (UFC)

Profa. Dra. Deborah Maria Vieira Magalhães
Universidade da Integração Internacional da Lusofonia Afro-Brasileira (UNILAB)

Prof. Dr. César Lincoln Cavalcante Mattos
Universidade Federal do Ceará (UFC)

Aos meus pais, que sempre se dedicaram para que minha graduação fosse tranquila e proveitosa. Esta conquista é nossa.

AGRADECIMENTOS

À Profa. Dra. Atslands Rego da Rocha e à Profa. Dra. Deborah M. V. Magalhães por me orientarem em meu trabalho de conclusão de curso, permitindo meu desenvolvimento acadêmico e profissional.

Aos meus amigos de curso Bruna, Carlim, Yohanne e Wildnei que fizeram da graduação uma jornada mais leve e prazerosa.

À minha namorada, Teresa, que me apoiou continuamente, sempre amenizando meus medos e ansiedades, e me fez acreditar que tudo daria certo.

Aos meus irmãos, Carol e Gabriel, que sempre me ouviram e aconselharam.

Aos meus pais, Sandra e Ponciano, sem os quais nada disso seria possível. À minha mãe, com seu otimismo inabalável, que nunca permitiu que eu desse voz às minhas inseguranças. Ao meu pai, sempre apoiando minhas escolhas.

À FUNCAP, pelo apoio financeiro, fundamental para a realização deste trabalho.

Agradeço a todos os professores que fizeram parte da minha trajetória acadêmica pelos aprendizados e pelo incentivo ao meu desenvolvimento pessoal e profissional.

“Mas vai se formar na faculdade primeiro. De graça, devo acrescentar. Fim de papo.”

(A Escolha Perfeita)

RESUMO

A plantação de milho é uma das lavouras mais comuns e economicamente pertinentes globalmente. Contudo, apesar da grande relevância econômica e nutricional, as plantações de milho ainda são afetadas pela praga *Spodoptera frugiperda* (J. E. SMITH, 1797) (LEPIDOPTERA: NOCTUIDAE). Para amenizar a problemática dessa praga, são adotados o monitoramento e o controle, como a verificação por meio de imagens capturadas por câmeras integradas às armadilhas e o controle biológico ou químico. Com o avanço da tecnologia, novos métodos de monitoramento para essas pragas foram propostos, como o uso do aprendizado profundo. Entretanto, essas técnicas não mitigam a problemática da privacidade de dados, que podem ter informações confidenciais e mercadológicas do campo do agricultor, e os desafios de cenários distribuídos, como a limitação de conectividade e de distribuição não homogênea dos dados entre usuários. O aprendizado federado propõe o treinamento descentralizado, em que não há compartilhamento de dados, ou seja, os dados ficam no dispositivo do usuário, garantindo maior privacidade. Neste trabalho é proposta uma abordagem de utilização do aprendizado federado com diferentes agregações, garantindo uma maior privacidade e menor envio de dados, minimizando o uso de recursos de rede. Além disso, há a simulação de cenários do campo de agricultura para avaliar quais métodos de agregação são promissores para cenários distribuídos. Utilizou-se uma base de dados contendo imagens capturadas por câmeras integradas à *Raspberry Pi* em armadilhas. Essas imagens foram divididas em duas classes: com a presença da *S. frugiperda* e sem a presença do inseto. Esse conjunto de dados foi distribuído de duas maneiras distintas, em vegetativo (caracterizado pelo número de colarinhos foliares visíveis) e em reprodutivo (marcado pelo desenvolvimento dos grãos), que representam a fase do milho, ambas não independentes e identicamente distribuídas. Os dois cenários foram testados com quatro algoritmos de agregação: FedAvg, FedAvgM, FedProx e FedYogi. Além disso, foi analisado o consumo de Unidade Central de Processamento (CPU) e de memória. Os resultados mostram que, em todos os cenários, o FedProx obteve os melhores resultados na maioria das métricas de avaliação de modelos, mas com ressalva quanto ao consumo de recursos computacionais. No cenário vegetativo, FedYogi demonstrou ser uma boa opção, consumindo menos recursos computacionais e apresentando bons resultados para as métricas de avaliação.

Palavras-chave: Aprendizado Federado. Método de Agregação. Classificação. Agricultura.

ABSTRACT

Corn planting is one of the most common and economically relevant crops globally. However, despite its great economic and nutritional importance, corn plantations are still affected by the pest *Spodoptera frugiperda* (J. E. SMITH, 1797) (LEPIDOPTERA: NOCTUIDAE). To mitigate the problems caused by this pest, monitoring and control strategies are employed, including the use of camera-equipped traps for image capture and the application of biological or chemical control methods. With advances in technology, new monitoring methods for these pests have been proposed, such as deep learning. Nevertheless, these techniques do not address data privacy issues, as the data may contain confidential and market-sensitive information from the farmer's field, nor do they address the challenges of distributed scenarios, such as limited connectivity and non-homogeneous data distribution among users. Federated learning proposes decentralized training, in which data is not shared, remaining on the user's device and ensuring greater privacy. This work proposes an approach using federated learning with various aggregations, ensuring greater privacy and reduced data transmission, while minimizing network resource utilization. Additionally, a simulation of agricultural field scenarios is used to evaluate which aggregation methods are promising for distributed scenarios. We used a database containing images captured by cameras integrated into Raspberry Pi in traps. We divided these images into two classes: with the presence of S. frugiperda and without the presence of the insect. This dataset was distributed in two distinct ways: vegetative (characterized by the number of visible leaf collars) and reproductive (marked by grain development), representing the corn growth stage, and both non-independent and identically distributed. We tested the two scenarios with four aggregation algorithms: FedAvg, FedAvgM, FedProx, and FedYogi. Furthermore, we analyzed CPU and memory consumption. The results show that, across all scenarios, FedProx achieved the best performance in most model evaluation metrics, though with reservations about computational cost. In the vegetative scenario, FedYogi proved to be a good option, consuming fewer computational resources and presenting good results for evaluation metrics.

Keywords: Federated Learning. Aggregation Method. Classification. Agriculture.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AP	Agricultura de Precisão
CNN	Rede Neural Convolucional
CPU	Unidade Central de Processamento
DL	Aprendizado Profundo
DNN	Rede Neural Profunda
ECA	Atenção Eficiente de Canal
FL	Aprendizado Federado
GB	Gigabyte
IID	Independentes e Identicamente Distribuídos
IoT	Internet das Coisas
MB	Megabyte
MCC	Coeficiente de Correlação de Matthews
ML	Aprendizado de Máquina
MLP	Perceptron Multicamadas
R-CNN	Redes Neurais Convolucionais Baseadas em Regiões
ReLU	Unidade Linear Retificada
RL	Aprendizado por Reforço
SGD	Gradiente Descendente Estocástico
UAV	Veículo Aéreo Não Tripulado
VM	Máquina Virtual

LISTA DE SÍMBOLOS

$f(X)$	Função da rede neural (saída do modelo)
α_0	Termo de intercepto (viés da saída)
α_k	Peso associado à k-ésima unidade oculta na saída
$h_k(X)$	Saída da k-ésima unidade oculta
A_k	Representação da ativação da k-ésima unidade oculta
$g(\cdot)$	Função de ativação (ex.: ReLU, sigmoid, tanh)
w_{k0}	Viés da k-ésima unidade oculta
w_{kj}	Peso da conexão entre a j-ésima entrada e a k-ésima unidade oculta
X_j	j-ésima variável de entrada (feature)
n	Número de variáveis de entrada
K	Número de unidades ocultas da camada escondida
β	Momento
μ	Hipermarâmetro atuante no peso da penalidade no termo proximal
z_t	Modelo global na rodada t
z_t^i	Parâmetros do modelo local do cliente i na rodada t
S_t	Subconjunto de clientes selecionados na rodada t
c_i	Número de exemplos de dados no cliente i
c	Número total de exemplos de dados
m_t	Soma dos exemplos de dados de todos os clientes em S_t
Δz	Atualização de peso agregada
v	Vetor de momento
$F_i(z)$	Função de perda empírica local do cliente i
$h_i(z;z_t)$	Função substituta no FedProx
I	Número de dispositivos/clientes selecionados
η	Taxa de aprendizado do servidor
γ	Parâmetro de decaimento

q_t	Primeiro momento na rodada t
d_t	Estimativa do segundo momento na rodada t
τ	Parâmetro de adaptabilidade

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização do Problema

A plantação de milho é uma das lavouras mais comuns e economicamente pertinentes globalmente, sendo um cereal de suma importância para a alimentação humana e animal. O Brasil, em 2023, ocupou a terceira posição na produção de milho, com uma produção correspondente a 10,6% do total global (Embrapa, 2025). Contudo, apesar de sua grande relevância econômica e nutricional, as plantações de milho ainda são intensamente afetadas por pragas, como a *Spodoptera frugiperda*, considerada uma das maiores pragas agrícolas. Essa praga afeta não somente culturas de milho, mas também outros tipos de lavouras, como o algodão, sendo uma peste altamente polífaga, causando grande impacto na produtividade. Economicamente, essa praga agrícola possui danos significativos, como ocorreu em doze países da África, onde houve uma perda de cerca de 2,5 a 6,3 bilhões de dólares em 2017 (Overton *et al.*, 2021). Tendo isso em vista, é imprescindível haver maior controle e monitoramento da *S. frugiperda*.

Historicamente, o controle da praga *Spodoptera frugiperda* era realizado, principalmente manualmente, exigindo maior esforço de tempo e mão de obra, sendo mais suscetível a erros humanos. Atualmente, com os avanços em recursos tecnológicos, houve um maior desenvolvimento de ferramentas baseadas em Aprendizado de Máquina (ML) (do inglês, *Machine Learning* - ML) e, mais recentemente, em Aprendizado Profundo (DL) (do inglês, *Deep Learning* - DL), que auxiliam no monitoramento. O uso de Rede Neural Convolucional (CNN) (do inglês, *Convolutional Neural Network* - CNN) automatiza processos e melhora significativamente a detecção de pragas agrícolas (Venkateswara; Padmanabhan, 2025). Além disso, outro avanço importante foi a integração de técnicas de ML e Internet das Coisas (IoT) (do inglês, *Internet of Things* - IoT), sendo possível embarcar modelos de redes neurais em aparelhos acessíveis economicamente, a exemplo do *Raspberry Pi* (Soares *et al.*, 2025). Apesar disso, o treinamento centralizado apresenta alguns desafios notórios, como a falta de segurança, já que a centralização dos dados em um único local pode torná-los mais suscetíveis à exposição e a ataques (Saif *et al.*, 2025). Ademais, possui menor escalabilidade e eficiência, pois com o aumento de dados, há o aumento de requisitos computacionais e de armazenamento, podendo exceder a capacidade da máquina, e assim a diminuição da eficiência (Saif *et al.*, 2025).

Esses grandes avanços na utilização de IoT para a classificação de pragas em campo de agricultura contribuíram significativamente para o controle de pragas no setor agrícola (Pra-

sath; Akila, 2023). Todavia, essa integração de modelos de ML com IoT enfrenta problemáticas importantes a serem analisadas. Sabe-se que o uso tradicional de DL apresenta altos custos de comunicação e armazenamento, sendo um grande desafio para dispositivos IoT que possuem capacidade computacional limitada. Além disso, esses modelos não performam bem em lavouras com dados insuficientes, desbalanceados e em meios complexos (Deng *et al.*, 2022), cenários comuns para o campo da agricultura.

Nesse contexto, o Aprendizado Federado (FL) (do inglês *Federated Learning* - FL) mostra-se uma alternativa propícia. O FL apresenta técnicas que possibilitam o treinamento colaborativo entre dispositivos de modelos de ML sem a precisão de compartilhamento de dados para mitigar questões, garantindo maior privacidade e efetividade de comunicação (Ramos *et al.*, 2021). A privacidade dos dados agrícolas é de suma importância, já que, com a falta dela, pode haver a violação de segredos de negócios dos agropecuaristas (Ejnisman *et al.*, 2019), podendo causar diversos impactos negativos no lucro desses profissionais. Além disso, a efetividade de comunicação se dá pelo envio apenas dos pesos dos modelos, ou seja, há redução das informações transmitidas.

Embora os avanços de IoT e ML na agricultura tenham sido pertinentes, ainda existem desafios no desempenho do FL, principalmente em cenários com dados não igualmente distribuídos (dados não-Independentes e Identicamente Distribuídos (IID)), em que os dados são distribuídos de maneira não uniforme entre os clientes (Jimenez G. *et al.*, 2024), ou seja, usuários podem possuir mais dados e tipos de classes do que outros. Nesse contexto, esse trabalho busca analisar e aplicar estratégias de FL presentes na literatura, propondo reduzir as problemáticas dos modelos centralizados voltados à classificação da *S. frugiperda* em campo de agricultura.

Além da contribuição tecnológica, este trabalho também demonstra importância social ao propor soluções mais seguras, pois o uso da tecnologia para a classificação da praga automatiza processos, diminuindo a exposição do agricultor a potenciais riscos à saúde a longo prazo. Ademais, a utilização de FL tem como objetivo a proteção de dados sensíveis, como dados relacionados à produção de plantas e dados de solo (Mendes *et al.*, 2023), e a ampliação da autonomia dos agricultores.

Os algoritmos de agregação são de suma importância para lidar com dados não-IID no FL, pois contribuem para maior estabilidade e melhor desempenho do modelo global. Nesse contexto, este trabalho contribui com a avaliação e comparação de estratégias de FL empregadas na classificação da *S. frugiperda* em ambientes que possuem dados não uniformes, em específico,

campo de agricultura.

1.2 Objetivos

Nessa seção são descritos os principais objetivos desse trabalho, que objetiva analisar e aplicar algoritmos presentes na literatura que buscam minimizar a problemática dos dados não-IID no campo de agricultura, em específico para identificação da *S. frugiperda*.

1.2.1 Objetivo Geral

O objetivo principal deste trabalho é aplicar as principais técnicas de agregação de FL presentes na literatura para comparação, considerando os recursos computacionais e as métricas de avaliação do desempenho dos modelos, além da distribuição de uma base de dados de forma heterogênea, que inclui imagens da *S. frugiperda*, obtidas por câmeras em campos de agricultura.

1.2.2 Objetivos Específicos

Para alcançar o objetivo geral, os seguintes objetivos específicos do trabalho são listados a seguir:

- Aplicar pelo menos dois tipos diferentes de agregação de FL;
- Separar dados de forma não-IID, simulando heterogeneidade;
- Utilizar contêineres para limitar o uso de recursos computacionais de cada cliente, simulando a utilização de dispositivos *Raspberry Pi*;
- Avaliar o desempenho dos diferentes métodos de agregação implementados.

1.3 Organização do Trabalho

Este trabalho está dividido em cinco capítulos principais:

- **Capítulo 2 - Trabalhos Relacionados:** São discutidos estudos e pesquisas anteriores, presentes na literatura, relevantes sobre FL e classificação de pragas agrícolas.
- **Capítulo 3 - Fundamentação Teórica:** Apresenta os conceitos essenciais para a compreensão do trabalho.
- **Capítulo 4 - Metodologia:** É descrito o procedimento utilizado para a realização dos experimentos, as principais ferramentas e as métricas utilizadas para avaliar os resultados.

- **Capítulo 5 - Resultados:** São apresentados os resultados dos cenários propostos, obtidos a partir dos experimentos adotados.
- **Capítulo 6 - Conclusão:** São abordados os objetivos centrais do trabalho e os principais resultados alcançados. Além disso, apresenta os limites desse estudo e aponta direções para trabalhos futuros no contexto do FL em campo de agricultura.

2 TRABALHOS RELACIONADOS

Nesta seção, são abordados e discutidos trabalhos da literatura relevantes ao contexto apresentado. Foram realizadas pesquisas entre 2022 e 2025, nas quais o tema abordado concentra-se principalmente na proposta de implementar FL em diferentes áreas da agricultura, como controle de pragas e doenças em plantas. As buscas por trabalhos científicos foram feitas em cinco bases de dados científicas distintas: *Google Scholar*, *Scientific Electronic Library Online* (*SciELO*), *ArXiv*, *IEEE Xplore* e *ScienceDirect*.

2.1 (Piccialli *et al.*, 2025) AGRIFOLD: AGRIculture Federated learning for Optimized Leaf disease Detection

Em (Piccialli *et al.*, 2025) os pesquisadores propuseram o AGRIFOLD, um *framework* de FL projetado para permitir o treino colaborativo de uma CNN leve, mantendo a privacidade, utilizando o mecanismo Atenção Eficiente de Canal (ECA) juntamente com a arquitetura VGG16. Nesse estudo, foram utilizadas 12 bases de dados, disponibilizadas gratuitamente, contendo imagens de plantas doentes, distribuídas heterogeneamente (não-IID) entre os clientes.

Além disso, a avaliação do modelo de FL foi realizada com base em testes de vários métodos de agregação, tais como FedAvg, FedProx, SCAFFOLD, FedBN e FedDF. Os resultados das agregações utilizadas foram avaliados pelas métricas de avaliação: acurácia, precisão, *recall* e *f1-Score*. O método SCAFOLD apresentou melhor desempenho, considerando a estabilidade da acurácia entre clientes e seu alto desempenho.

Diferentemente do AGRIFOLD, este trabalho utiliza, como métodos de agregação, além de FedAvg e FedProx, os algoritmos FedAvgM e FedYogi. Os resultados são avaliados com base nas métricas obtidas pelo modelo global, sendo elas acurácia, Função Perda, F1-score e Kappa. Além disso, ambos os trabalhos utilizam uma distribuição não-IID no contexto do FL.

2.2 (Tanwar *et al.*, 2025) ReinDSplit: Reinforced Dynamic Split Learning for Pest Recognition in Precision Agriculture

O trabalho dos autores (Tanwar *et al.*, 2025) adota uma abordagem diferente de FL, mantendo ainda a propriedade distribuída dessa arquitetura. Os pesquisadores introduzem o ReinDSplit, uma estrutura nova baseada em Aprendizado por Reforço (RL) que dinamicamente adapta os pontos de divisão da Rede Neural Profunda (DNN) para cada dispositivo. Além

disso, utiliza-se um agente *Q-learning* que age como um orquestrador adaptativo que balanceia *workloads* e limites de latência pelos dispositivos para mitigar a escassez ou sobrecarga computacional.

Nessa pesquisa, foram utilizados três conjuntos de dados públicos de pragas, com diferentes classes, distribuídos tanto heterogeneamente (IID) quanto não heterogeneamente (Não-IID), testados em três modelos distintos: ResNet18, GoogleNet e MobileNetV2. Esses experimentos foram avaliados por 5 métricas, sendo a acurácia a principal métrica de avaliação.

Comparando este trabalho com a pesquisa de (Tanwar *et al.*, 2025), observam-se diferenças relevantes no reconhecimento de pragas no contexto agrícola. A principal diferença deste trabalho reside no uso da abordagem de FL e de diferentes métodos de agregação para a avaliação do modelo. No entanto, no estudo de (Tanwar *et al.*, 2025) utiliza-se *ReinDSPlit* que também se baseia em aprendizado distribuído. Nesse trabalho, os principais métodos de avaliação são acurácia, precisão, Recall, F1-score e Coeficiente de Correlação de Matthews (MCC). Além disso, este trabalho tem como objetivo a classificação específica de um tipo de praga em campo de agricultura, a *Spodoptera frugiperda*. Já no trabalho de (Tanwar *et al.*, 2025), apesar de existir, em seu conjunto de dados, uma praga que afeta plantações de milho, a broca-do-milho europeia, não há a presença da *S. frugiperda*.

2.3 (Ahmed *et al.*, 2022)Federated learning-based UAVs for the diagnosis of Plant Diseases

No trabalho de (Ahmed *et al.*, 2022) é apresentada a classificação de diferentes pragas utilizando Veículo Aéreo Não Tripulado (UAV)s baseada em FL, com agregação FedAvg, em quatro campos diferentes, utilizando um conjunto de dados público com diversos tipos de pragas variadas, incluindo a *S. frugiperda*.

Em (Ahmed *et al.*, 2022), são analisadas quatro fazendas distintas, em que cada uma possui um UAV que analisa em tempo real e detecta pragas utilizando a arquitetura DL EfficientNet com configuração B03. Cada cliente coleta novos dados, imagens, em que pode haver desbalanceamento entre clientes, por isso, não-IID. Para a avaliação da arquitetura proposta, foi utilizada a acurácia.

A pesquisa de (Ahmed *et al.*, 2022) em comparação com este trabalho apresenta diferenças evidentes, apesar de possuírem objetivos semelhantes, como a aplicação do FL. Enquanto (Ahmed *et al.*, 2022) emprega UAV para a aquisição de dados, este trabalho utiliza câmeras integradas às armadilhas no lugar de UAV para a captura de imagens de pragas. Além

disso, este trabalho utiliza um conjunto de dados próprio e tem o objetivo de classificar apenas um tipo de peste agrícola, no caso, a *S. frugiperda*. Também, neste trabalho, são aplicados outros tipos de agregações para avaliar suas eficiências em cenários Não-IID, como o FedAvgM, FedProx e FedYogi, em que são utilizados outros tipos de métricas de avaliação além da acurácia, sendo elas Função Perda, Kappa e F1-score.

2.4 (Deng *et al.*, 2022) Multiple Diseases and Pests Detection Based on Federated Learning and Improved Faster R-CNN

Em (Deng *et al.*, 2022) é sugerida uma técnica de classificação de múltiplas pragas baseada em FL em conjunto com R-CNN otimizada e ResNet101. Foram utilizados seis conjuntos de dados diferentes que continham amostras de doenças e pragas agrícolas distintas, nos quais esses dados foram distribuídos não heterogeneamente entre os clientes, ou seja, de forma Não-IID.

Ademais, no trabalho de (Deng *et al.*, 2022) foi empregada a agregação FedAvg personalizada, na qual foi atribuído um parâmetro de restrição M para garantir a convergência do modelo e melhorar a velocidade de treino. Para avaliar a arquitetura proposta, foram aplicadas as métricas: acurácia, mAP, tempo de detecção, em que a acurácia alcançou 90,27%, mAp 89,34% e a velocidade de treino foi melhorada em 59%.

Diferentemente do estudo de (Deng *et al.*, 2022), este trabalho foca apenas na classificação *S. frugiperda* e também utiliza outros tipos de agregação de FL além da FedAvg, como o FedAvgM, FedProx e FedYogi, no caso do trabalho comparado, um FedAvg personalizado. Já nas métricas de avaliação, o estudo de (Deng *et al.*, 2022) utiliza o tempo de detecção, acurácia e mAp, o que difere deste trabalho, que utiliza Função Perda, F1-score, acurácia e kappa.

2.5 Conclusão

É perceptível, após a análise dos trabalhos relacionados, que a maioria utilizou FedAvg como estratégia de agregação para analisar o FL, considerando a acurácia como uma das principais métricas de avaliação das arquiteturas propostas. Além disso, mais da metade dos conjuntos de dados consistia em imagens distribuídas desigualmente entre os clientes. As arquiteturas dos modelos foram diversas.

Portanto, o Quadro 1 sintetiza as lacunas fundamentais que este trabalho desejou

preencher, comparado com trabalhos relacionados, em que trabalhos anteriores, apesar de apresentarem semelhanças em alguns aspectos, não possuem o mesmo objetivo que este trabalho. Essa tabela resume os principais pontos que serão abordados. Em Objetivo, apresenta-se o principal objetivo do trabalho; em Dados, explicita-se o tipo de base de dados utilizada; em Arquitetura, evidencia-se as arquiteturas de modelos que foram implementadas; em Métricas, demonstra-se as métricas de avaliação utilizadas para analisar os resultados; em Agregação, são apontadas quais foram as técnicas de agregação testadas; e, em Distribuição, especifica-se o tipo de distribuição foi feita no trabalho.

Quadro 1 – Características científicas e analíticas de trabalhos relacionados à agricultura

Autores	**Objetivo**	**Dados**	**Arquitetura**	**Métricas**	**Agregação**	**Distribuição**
(Piccialli *et al.*, 2025)	*Framework* de FL otimizado projetado para detecção de doenças em folhas.	12 bases de dados públicos de imagens de doenças de plantas	VGG16 + ECA	Acurácia, Precisão, Recall, F1-Score	FedAvg, FedProx, SCAFFOLD, FedBN, e FedDF	Não-IID
(Tanwar *et al.*, 2025)	Estrutura que adapta dinamicamente pontos de divisão DNN para classificação de pragas	Três bases de dados de pragas	ResNet18, GoogleNet e MobileNetV2	acurácia, precisão, *recall*,*F1-score* e MCC	Não se Aplica	IID e Não-IID
(Ahmed *et al.*, 2022)	Classificação de pragas utilizando UAVs baseado em FL	Imagens de diferentes tipos de pragas	CNN customizada com EfficientNet-B3	Acurácia	FedAvg	Não-IID
(Deng *et al.*, 2022)	Técnica de detecção de múltiplas pragas baseada em FL	Imagens de maçãs doentes e com pragas	*Faster* R-CNN otimizado com ResNet-101	Acurácia, mAP e tempo de detecção	FedAvg personalizado	Não-IID
Este trabalho	**FL para classificação da *S. frugiperda* em campo de agricultura.**	**Base de dados obtidas a partir do trabalho de (Silva *et al.*, 2024)**	**Mobilenet + MLP**	**Acurácia, F1-Score, Kappa e Função Perda (Entropia Cruzada Binária)**	**FedAvg, FedAvgM, FedProx e FedYogi**	**Não-IID**

Fonte: elaborado pelo autor (2025).

3 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados conceitos teóricos básicos para um entendimento mais assertivo do trabalho realizado. São abordados a agricultura de precisão e o monitoramento de pragas agrícolas, o aprendizado de máquina e seus subtópicos relacionados, e o aprendizado federado.

3.1 Agricultura de precisão

A Agricultura de Precisão (AP) é uma estratégia de gerenciamento para atender à variabilidade temporal e geográfica em campos de agricultura, que envolve decisões voltadas à análise de dados e tecnologias modernas, com o objetivo de melhorar a eficiência de recursos, o que resulta na redução de custos enquanto diminui os impactos ambientais da agricultura (Karunathilake *et al.*, 2023). Consiste na aplicação de tecnologias de agricultura de precisão, baseadas em detecção, observação e resposta para administrar variabilidades espaciais e temporais para aperfeiçoar o desempenho de culturas e a qualidade ambiental, muito empregadas para monitorar infestações de pragas, por exemplo (Monteiro *et al.*, 2021).

3.1.1 Controle de pragas agrícolas

O controle de pragas agrícolas consiste em estratégias para conter a proliferação dessas, que assolam plantações, e reduzir os danos causados por elas, como o dano econômico. Existem diversas formas de controle são algumas delas: controle químico, com a utilização de agrotóxicos; controle natural, em que é realizado um conjunto de práticas culturais para minimizar o surgimentos de pragas, como poda sanitária e erradicação de plantas doentes; controle biológico, que consiste no controle a partir, por exemplo da liberação proposital de predadores naturais; e controle alternativo, que podem ir de armadilhas até uso de produtos caseiros (Embrapa, 2005).

No entanto, essas técnicas de controle citadas anteriormente apresentam desafios notórios, como efeitos negativos à saúde humana e ao meio ambiente decorrentes do uso de agrotóxicos e do excesso de trabalho necessário para realizar o controle manual, tanto no controle natural quanto no controle alternativo (Gill *et al.*, 2023). Além disso, apesar de no controle alternativo haver uso de armadilhas, o uso destas sem automatização pode, também, gerar um trabalho excedente para os agrônomos e agricultores.

As armadilhas são amplamente utilizadas no controle de pragas agrícolas, possibilitando o monitoramento das populações de insetos. A captura dos insetos em armadilhas é feita utilizando atrativos alimentares, como descrito em (Praxedes Junior, 2024), ou com a utilização de feromônios para atração dessas pragas. Entretanto, quando realizado de maneira convencional, o controle por armadilhas requer inspeção frequente, aumentando o tempo de trabalho e esforço do agricultor.

Por isso, o uso de ML no controle de pragas agrícolas é de suma importância, pois diminui a perda de colheitas (Gill *et al.*, 2023). Ademais, os agricultores podem aplicar o ML para a identificação precoce de pragas, diminuindo os custos de produção (Shah *et al.*, 2022). Além disso, com o FL esse tipo de controle se torna ainda mais benéfico, já que irá garantir a privacidade dos usuários e a personalização de dados.

Outrossim, o controle de pragas, como a *S. frugiperda*, deve ser realizado de forma mais assertiva em determinadas fases da cultura afetada, uma vez que a ocorrência do inseto se altera ao longo do ciclo fenológico. Um exemplo é observado na cultura do milho. Nesse cultivo, há dois estágios fenológicos principais, sendo eles: vegetativo e reprodutivo. A fase vegetativa é marcada pela maior presença da *S. frugiperda* havendo cerca de 3 vezes mais incidência dos insetos em relação à fase reprodutiva (Praxedes Junior, 2024).

O estágio reprodutivo tem início após o surgimento da espiga, o qual é determinado pelo desenvolvimento dos grãos (Bayer Crop Science, 2025). Já o estágio vegetativo é caracterizado pelo número de colarinhos foliares visíveis (Iowa State University Extension and Outreach, 2025). Esses colarinhos foliares são estruturas visíveis na planta. Antes de alcançar a altura máxima, a planta completa o estágio vegetativo e muda para o estágio reprodutivo (Bayer Crop Science, 2025).

3.2 Aprendizado de máquina

ML é uma subárea da Inteligência Artificial, a qual possibilita que um sistema aprenda e aprimore o reconhecimento de padrões de forma autônoma a partir de uma coleção de dados. O ML se concentra na construção de algoritmos para que máquinas possam aprender e aperfeiçoar-se à medida que novas informações são incorporadas, possibilitando a criação de modelos que possam realizar determinadas tarefas, como a detecção de anomalias (Oracle Corporation, 2025). O aprendizado de máquina consegue desvendar como realizar tarefas importantes a partir da generalização de exemplos (Domingos, 2012).

O ML pode ser dividido em quatro categorias de aprendizado, sendo elas: supervisionado, não supervisionado, semi-supervisionado e aprendizado por reforço.

1. Aprendizado supervisionado:

   No aprendizado supervisionado, o conjunto de dados é uma coleção de exemplos que são rotuladas $\{(x_i, y_i)\}_{i=1}^{N}$, onde cada elemento de x_i a N é chamado de vetor de características, ou vetor de *features* (Burkov, 2019). As características são as variáveis de entrada, também chamadas de preditores (James *et al.*, 2023). E funcionam da seguinte maneira: se levar em consideração um conjunto de dados de diferentes tipos de pragas agrícolas, o primeiro preditor x^1 pode ser a cor da praga e x^2 pode ser o tamanho em centímetros. Esses atributos, como cor e tamanho, compõem o vetor de características. Já y_i chamado de rótulo, ou *label*, é um elemento pertencente a um finito conjunto de classes $1, 2..C$, no qual y_i também pode ser uma estrutura mais complexa. Portanto, o rótulo representa o valor verdadeiro de cada exemplo. Por exemplo, os rótulos podem ser $lagarta-do-cartucho, mosca-branca, gangrena-da-batata$ (Burkov, 2019). Tendo isso em vista, o algoritmo de aprendizado supervisionado tem como objetivo o uso do conjunto de dados para produzir um modelo que recebe um vetor de características x como entrada e recebe uma predição a partir das informações de entrada (Burkov, 2019).

2. Não supervisionado:

   No aprendizado não supervisionado, trabalha-se com conjuntos de dados que não são rotulados, sendo constituídos por vetores de características $\{x_i\}_{i=1}^{N}$ (Burkov, 2019). Diferentemente do supervisionado, não recebe o y_i, variável de saída, como *label* (James *et al.*, 2023). Nesse contexto, o algoritmo busca reconhecer padrões ou relações entre as variáveis ou entre as observações (James *et al.*, 2023). O modelo recebe as entradas x e transforma-as em outro vetor ou em um valor que pode ser utilizado para resolver um problema (Burkov, 2019).

3. Semi-supervisionado:

   O aprendizado semi-supervisionado, vai ter características de ambos os aprendizados citados anteriormente (supervisionado e não supervisionado), ou seja, utiliza um conjunto de dados com exemplos rotulados e não rotulados (Burkov, 2019). O objetivo do aprendizado semi-supervisionado é semelhante ao supervisionado, em que se espera, usando exemplos não rotulados, que o algoritmo de aprendizado consiga achar um modelo de

melhor desempenho. Vale ressaltar que a quantidade de exemplos não rotulados, em geral, é significativamente superior à quantidade de exemplos que possuem rótulos (Burkov, 2020).

4. Aprendizado por reforço:

   O aprendizado por reforço é uma subárea do aprendizado de máquina na qual o sistema interage com o ambiente, percebendo o estado desse meio como um vetor de características. O objetivo desse aprendizado é determinar uma política ótima (*optimal policy*). Em que *policy* pode ser definida como uma função que recebe um vetor de características de um estado como entrada e tem como saída uma ação otimizada para executar nesse estado (Burkov, 2019), essa ação escolhida de forma a maximizar o desempenho. Ou seja, nesse algoritmo de aprendizagem, o modelo aprende a tomar decisões otimizadas por erro e tentativa, recebendo *feedback* de cada ação para alcançar melhores resultados (Amazon Web Services, 2024).

3.2.1 Redes Neurais

As redes neurais são modelos de ML inspirados, de modo abstrato, no funcionamento do cérebro humano, compostos por nós, ou também chamados de neurônios artificiais (*perceptrons*), organizados em uma arquitetura de camadas que são interconectadas (Amazon Web Services, 2025). Em uma rede neural simples vão existir três camadas, a camada de entrada (*input layer*), a camada oculta (*hidden layer*) e a camada de saída (*output layer*), como mostra a Figura 1.

Fundamentalmente, existem três tipos de classes de arquitetura de rede (Haykin, 2009):

1. *Single-Layer Feedforward Networks*

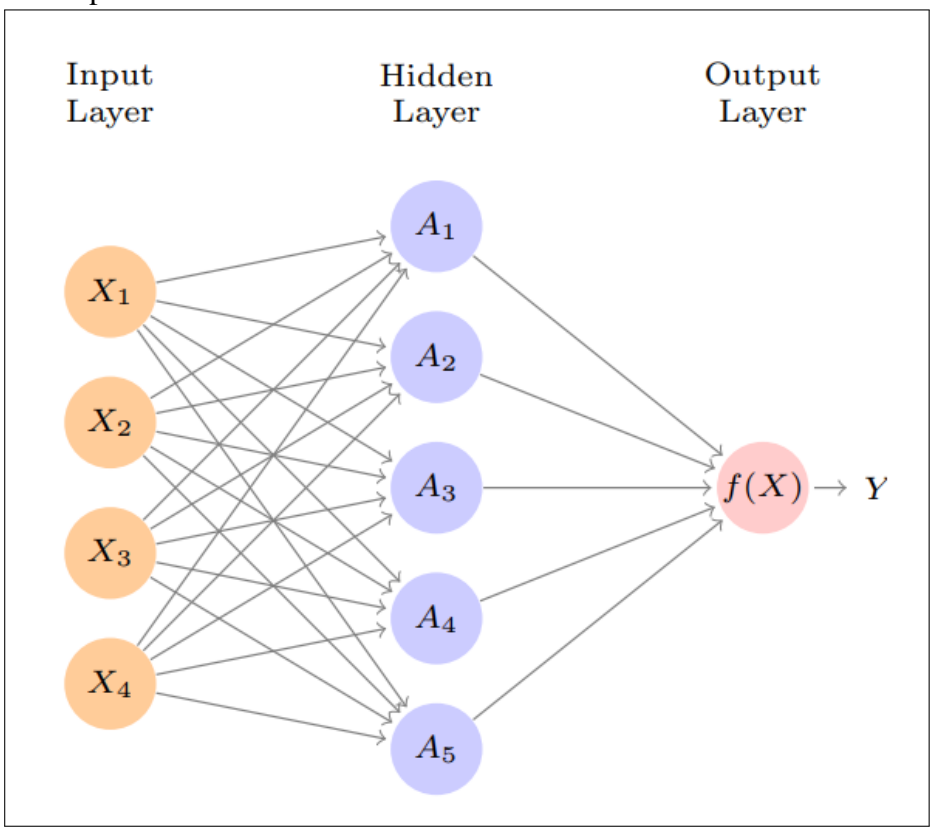
   Nesse tipo de arquitetura, existe uma camada de entrada que se conecta diretamente aos neurônios da camada de saída, mas o inverso não pode acontecer. É denominada de tipo *feedforward* justamente por essa característica (Haykin, 2009).

2. *Multilayer Feedforward Networks*

   Nessa segunda classe de arquitetura, é caracterizada pela presença de uma ou mais camadas ocultas, exemplificada na Figura 1, em que os nós nessa camada são chamados de neurônios ocultos ou unidades ocultas (Haykin, 2009).

3. *Recurrent Networks*

Figura 1 – Exemplo de Redes Neurais



Fonte: (James *et al.*, 2023).

O que diferencia essa arquitetura das arquiteturas citadas anteriormente é a presença de *loop* de realimentação, que pode ser um ou mais *loops*. Como dito previamente, em uma rede neural simples (*Single-Layer Feedforward Networks*), cada neurônio de entrada envia sua saída para as entradas de outros neurônios, mas não manda para o próprio neurônio, ou seja, sem *feedback*, e não possui camadas ocultas (Haykin, 2009). Já em uma *recurrent network*, mesmo que possua apenas uma camada de neurônios, cada neurônio pode enviar o sinal de saída de volta para a entrada dos demais neurônios da camada (Haykin, 2009).

Um exemplo básico, fornecido por (James *et al.*, 2023), de como uma dessas arquiteturas (*Single Layer*) pode ser caracterizado como um modelo que recebe um conjunto de variáveis de entrada, estruturado na camada de entrada (*input layer*), em um vetor $X = (X_1, X_2, X_3, \ldots, X_n)$ com n variáveis e produz uma resposta Y, obtida pela camada de saída (*output layer*), fornecida por combinações não lineares, gerando a função não linear $f(X)$ (James *et al.*, 2023). As setas indicadas na Figura 1 mostram a conexão de cada camada e cada uma das K unidades. O modelo de rede neural segue a seguinte forma (James *et al.*, 2023):

$$f(X) = \alpha_0 + \sum_{k=1}^{K} \alpha_k h_k(X) = \alpha_0 + \sum_{k=1}^{K} \alpha_k g\left(w_{k0} + \sum_{j=1}^{n} w_{kj} X_j\right), \tag{3.1}$$

As k's ativações A_k, onde $k = 1, ..., K$, correspondentes às unidades da camada oculta, são obtidas a partir de funções das variáveis $X_1, ..., X_n$, em que $g(\cdot)$ é a função de ativação que é normalmente definida previamente pelo desenvolvedor durante a etapa de projeto do modelo (James *et al.*, 2023). Cada variável A_k pode ser interpretada como diferentes transformações $h_k(X)$ da variável original, e as K ativações da camada oculta são combinadas na camada de saída, o que, assim, resulta em uma regressão linear. Todos os parâmetros $\alpha_0, \ldots, \alpha_K$ e $w_{10}, \ldots, w_{Kn}$ precisam ser estimados pelos dados (James *et al.*, 2023).

$$A_k = h_k(X) = g\left(w_{k0} + \sum_{j=1}^{n} w_{kj} X_j\right), \quad k = 1, \ldots, K. \tag{3.2}$$

Além disso, as funções de ativação desempenham papel crucial na rede neural, onde aprendem características abstratas a partir de transformações não-lineares (Dubey *et al.*, 2022). Uma das mais conhecidas funções de ativação são:

1. *Sigmoid*

   Nessa função de ativação, limita as saídas entre $[0, 1]$ (Dubey *et al.*, 2022). Essa função de ativação possui um gráfico característico em formato de "S"e segue a equação abaixo:

   $$LogisticSigmoid(x) = \left(1/1 + e^{-x}\right) \tag{3.3}$$

2. *Unidade Linear Retificada*

   A função de ativação Unidade Linear Retificada (ReLU) é uma das mais utilizadas nos problemas de *DL*. Ela funciona adicionando um limiar nos valores de saída: $f(x) = max(0, x)$. Simplificando, as saídas possuem valor 0 quando $x < 0$ e se $x >= 0$ têm como saída uma função linear (Agarap, 2019).

3.2.1.1 Redes neurais convolucionais

Uma CNN é um tipo de rede neural artificial *feedforward*, projetada para detectar padrões bidimensionais com elevado grau de invariância à translação e a certas formas de distorção (Haykin, 2009). A CNN utiliza dados de duas dimensões, nas quais explora filtros convolucionais que capturam certos padrões. Essa detecção é feita, geralmente, de forma

supervisionada com uma rede estruturada e segue os seguintes componentes fundamentais (Haykin, 2009):

1. Extração de características:

   Cada nó recebe sua respectiva entrada sináptica de um campo receptivo na camada anterior, forçando a extrair características locais (Haykin, 2009). Portanto, a partir do momento em que a características foi extraída, sua localização exata se torna menos importante, contanto que haja preservação de sua posição relativa a outras caraterísticas. Ou seja, após a detecção de padrão, a posição exata já não é tão relevante (Haykin, 2009).

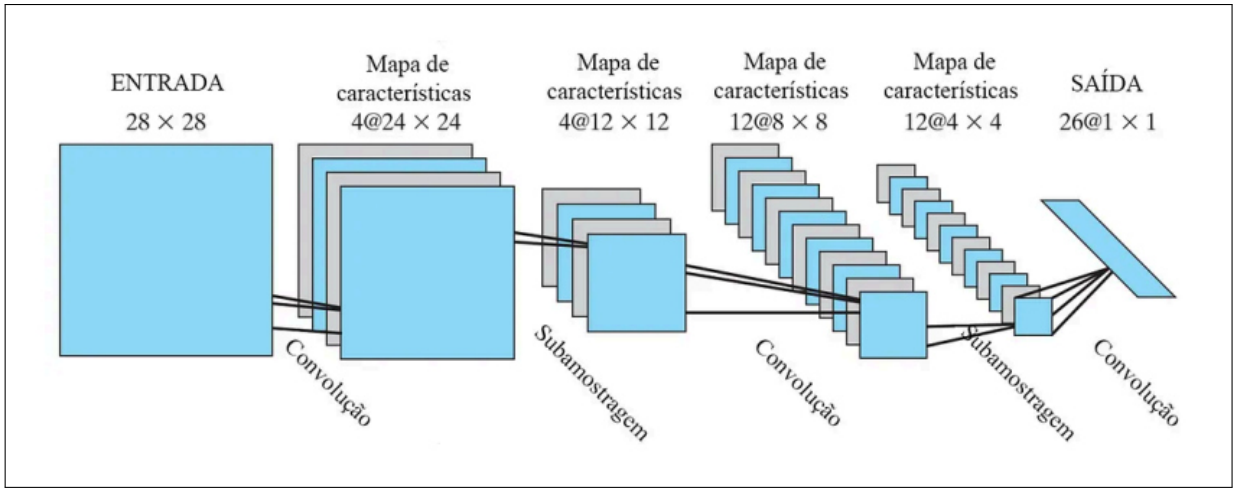
2. Mapa de características:

   Toda camada computacional da rede é composta por vários mapas de características. Cada mapa de caraterística tem um plano em que cada neurônio deve compartilhar o mesmo grupo de pesos sinápticos; os mesmos filtros devem ser utilizados em toda a imagem (Haykin, 2009). Cada célula no mapa de características corresponde a um grupo de *pixels* vizinhos, por exemplo, e, com isso, ajuda a ter vantagem em qualquer correlação que possa existir no dado (Haykin, 2009).

3. Subamostragem:

   Em cada camada convolucional sucessiva de uma camada computacional, executam-se operações de subamostragem, a exemplo de *max pooling* e *average pooling*, com objetivo de reduzir a dimensão dos mapas de características (Haykin, 2009). Esses passos fazem com que ocorra a redução de sensibilidade da saída do mapa de característica em relação a deslocamentos e outras formas de distorção que possam ocorrer (Haykin, 2009). Ou seja, reduz o número de parâmetros nas funções de ativação (O'Shea; Nash, 2015).

Figura 2 – Exemplo de Arquitetura de Rede Neural Convolucional



Fonte: (Poloni, 2022).

A Figura 2, originada de (Haykin, 2009), mostra um exemplo de arquitetura de uma CNN constituída de uma camada de entrada, quatro camadas ocultas e uma camada de saída. Esse exemplo simula o reconhecimento de caracteres escritos à mão. A camada de entrada recebe as imagens de dígitos, que foram previamente centralizadas e normalizadas, e é formada por uma matriz de 28 x 28 neurônios (Haykin, 2009).

Assumindo como exemplo a arquitetura proposta por (Haykin, 2009), na primeira camada oculta, responsável por armazenar valores dos *pixels* da imagem (O'Shea; Nash, 2015), ocorre a primeira convolução, que é responsável por extrair características. Em cada neurônio há uma análise de uma região da imagem, com campo receptivo de tamanho 5 x 5 (Haykin, 2009). Na segunda camada oculta, são executadas operações de subamostragem e *averaging* local (Haykin, 2009). Cada neurônio dessa camada tem um campo receptivo de tamanho 2 x 2, além de coeficiente e *bias* treináveis, juntamente com uma função de ativação do tipo *sigmoid*. Outro exemplo de função de ativação muito utilizado atualmente é ReLU. O coeficiente e o *bias* controlam o ponto de operação do neurônio. Nessa camada a resolução é reduzida, com isso, a rede se torna menos sensível a variações sutis (Haykin, 2009). Na terceira camada oculta, ocorre outra convolução, nela vão ter 12 mapas de características com tamanho de 8 x 8 neurônios, nos quais cada neurônio pode receber entradas provenientes de múltiplos mapas da camada anterior (Haykin, 2009). Na quarta e última camada oculta acontecem a segunda subamostragem e *averaging* local, possuindo 12 mapas de características organizados em matrizes de dimensão 4 x 4 de neurônios. Na última camada, camada de saída, acontece também a última convolução (Haykin, 2009). Essa camada vai ter 26 neurônios, cada neurônio representando uma letra do alfabeto (Haykin, 2009).

Esse exemplo da Figura 2 demonstra que a rede neural alterna entre convolução e subamostragem e, como resultado, tem uma saída compacta e classificatória.

Outrossim, vale ressaltar que o exemplo descrito anteriormente segue principalmente a formulação clássica de CNN descrita por (Haykin, 2009). Arquiteturas modernas podem diferir, em geral, quanto à utilização de funções de ativação e subamostragem, por exemplo.

3.3 Aprendizado Federado

Sabe-se que o ML clássico é notório pela realização de diferentes tarefas, como a detecção de objetos em imagem. Contudo, esse modelo de aprendizado centralizado apresenta desafios importantes, principalmente em muitos cenários reais. Alguns desses desafios são:

1. Regulamentação:

   Em muitos locais, essas regulamentações podem impedir que organizações combinem os dados dos seus clientes para treinamento de ML, já que muitos vivem em diferentes países que são regidos por diferentes regulamentações de proteção de dados, como o Brasil, que é regido pela LGPD (Flower Labs, 2025).

2. Preferência de usuário:

   Em alguns cenários, os usuários apenas esperam que os seus dados não saiam de seus dispositivos pessoais, como se um usuário digitasse seu número de cartão no teclado do celular, ou até o histórico de localização, não é esperado que esses dados apareçam no servidor da empresa (Flower Labs, 2025).

3. Volume de dados:

   Alguns sensores, como é o caso de câmeras, geram uma grande quantidade de dados, tornando inviável e economicamente custoso a coleta de todos os dados (Flower Labs, 2025).

4. Falta de memória:

   Os modelos clássicos de ML usualmente são treinados, sendo carregadas amostras de treinamento inteiramente na memória principal da máquina (Ramos *et al.*, 2021). Com isso, caso a complexidade computacional dos dados exceda a memória principal, pode causar diversos problemas no treinamento, como o baixo desempenho (Ramos *et al.*, 2021).

5. Tempo de treino inviável:

   Em certos processos de otimização usados nos algoritmos de ML pode não resultar em uma boa escalabilidade em relação ao tamanho do número de amostras (Ramos *et al.*, 2021). À medida que o tamanho do conjunto de dados aumenta, podem-se tornar inviáveis as horas utilizadas para treinar com esse conjunto de dados (Ramos *et al.*, 2021).

O FL surgiu como necessidade, propondo garantir que os dados de treino fiquem nos dispositivos do usuário e facilitar modelos complexos de ML colaborativo em dispositivos distribuídos (Ramos *et al.*, 2021).

De maneira simplificada, o FL funciona nas seguintes etapas (Flower Labs, 2025):

1. Inicialização do modelo global:

   Começa inicializando o modelo no servidor, muito semelhante ao sistema centralizado; os parâmetros são inicializados.

2. Envio do modelo para um conjunto de organizações ou dispositivos conectados chamados

de clientes:

Os parâmetros do modelo global são enviados para os clientes conectados, garantindo que cada cliente participante inicie o treino local utilizando o mesmo parâmetro do modelo global. Normalmente, nem todos nós conectados são utilizados, já que muitas vezes o uso de muitos clientes resulta em retornos decrescentes.

3. Treino do modelo local com dados de cada organização ou dispositivos:

   Após todos os clientes possuírem a última versão dos parâmetros do modelo global, eles utilizam o próprio conjunto de dados local para treinar o próprio modelo. Geralmente, o treino não acontece até a convergência completa, como no treinamento centralizado, mas apenas por um tempo determinado, por exemplo, pela quantidade de épocas locais, que pode ser uma, ou pela quantidade de *batches*.

4. Retorno da atualização do modelo local para o servidor:

   Depois do treino local, cada cliente tem uma versão diferente dos parâmetros do modelo. Esses parâmetros são diferentes justamente por terem sido treinados com diferentes conjuntos de dados. Esses parâmetros locais atualizados são mandados de volta para o servidor. As atualizações do modelo podem ser tanto todos os parâmetros quanto apenas gradientes que foram acumulados durante o treino local .

5. Atualização do modelo de agregação em novo modelo global:

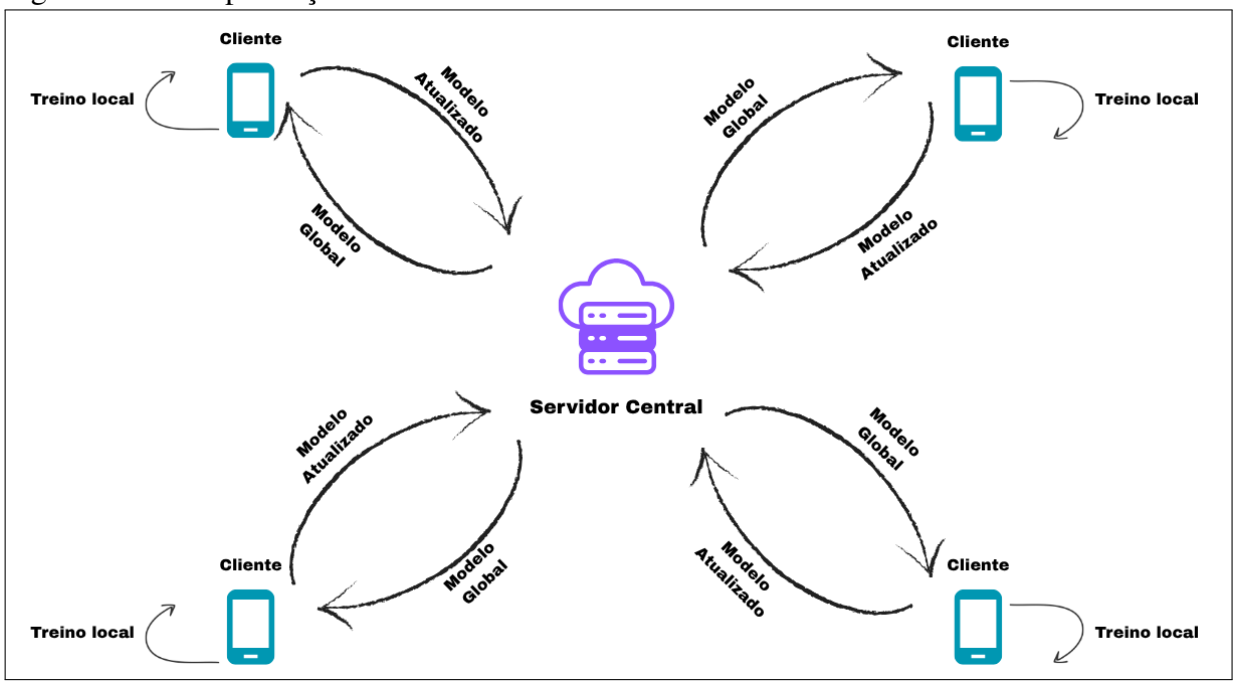
   Nessa etapa, o servidor central recebe as atualizações do modelo de cada cliente selecionado. Para obter apenas um modelo, é preciso combinar todas as atualizações de modelo recebidas dos clientes. Esse processo de combinação é chamado de agregação, na qual existem diferentes formas de acontecer, como a agregação *FedAvg*. Essa agregação *FedAvg*, chamada de *Federated Averaging*, proposta por (McMahan *et al.*, 2017), é a mais básica das agregações.

6. Repetição dos passos 2 a 5 até convergência do modelo:

   Os passos 2 a 5 são chamados de rodada; eles representam uma única rodada no processo do FL. É preciso repetir o processo de treinamento até alcançar um modelo totalmente treinado que consiga desempenhar bem em todos os conjuntos de dados de todos os clientes.

Essas etapas resumem o funcionamento do FL de maneira simplificada, também demonstrado na Figura 3, levando em consideração as principais etapas do treinamento e como o modelo é atualizado a partir de cada cliente.

Figura 3 – Exemplificação do funcionamento do FL



Fonte: Elaborado pela autora (2025).

3.3.1 Tipos de agregação

Como dito anteriormente, no FL existem combinações dos modelos recebidos pelos nós de clientes, e existem diferentes tipos de agregação e como são aplicados. A *Federated Averaging* é a mais comum e básica das agregações, mas existem diversas outras, que muitas vezes são variantes da *FedAvg*. A FedProx é um exemplo que propõe resolver problemas específicos, como é o caso de Dados Não-IID, que são dados não igualmente distribuídos entre clientes. Como exemplo de dados não-IID, se existem 10 clientes e os 5 primeiros recebem 90% da classe 1 e 20% da classe 0, enquanto os 5 últimos recebem 10% da classe 1 e 80% da classe 0. Já nos dados *IID*, as classes são distribuídas igualmente entre os clientes.

3.3.1.1 FedAvg

O FedAvg, ou *FederatedAveraging*, surgiu como uma maneira de mitigar a problemática de dados não-IID e também propõe a diminuição da utilização de recursos computacionais (McMahan *et al.*, 2017). Esse algoritmo utiliza a técnica de Gradiente Descendente Estocástico (SGD) para a otimização em conjunto com a média local dos modelos. O SGD pode ser aplicado ao problema de otimização do FL, em que, em cada *batch*, o cálculo do gradiente é realizado em cada rodada de comunicação. Essa abordagem é computacionalmente eficiente, porém, é necessária uma maior utilização do número de rodadas globais de treino para obter bons modelos

(McMahan *et al.*, 2017). A equação é exemplificada abaixo:

$$z_{t+1} \leftarrow \sum_{i \in S_t} \frac{c_i}{m_t} z_{t+1}^i \tag{3.4}$$

Onde z_{t+1} é o novo modelo global atualizado depois da rodada de treinamento. S_t é o subconjunto de clientes selecionados para participar da rodada específica. z_{t+1}^i são os parâmetros do modelo, resultado do treinamento local no cliente i. c_i é o número de exemplos de dados existentes no cliente i. Por último, m_t é a soma dos exemplos de dados de todos os clientes escolhidos no conjunto S_t.

3.3.1.2 FedAvgM

Proposto no trabalho de (Hsu *et al.*, 2019), o FedAvgM *(Federated Averaging with Server Momentum)* propõe a melhoria na métrica de acurácia em cenários com distribuição de dados não homogênea. No FedAvgM há uma alteração do FedAvg padrão em relação à atualização dos pesos. Nesse novo método de agregação, é adicionado o *momentum* (β) na parte do servidor (Hsu *et al.*, 2019). Seguindo a equação:

$$\Delta z = \sum_{i=1}^{I} \frac{c_i}{c} \Delta z_i \tag{3.5}$$

Sendo Δz a atualização de peso pelo cliente i, c_i o número de exemplos no cliente i, e c o número total de exemplos. Como padrão, o FedAvg atualiza os pesos z utilizando $z \leftarrow z - \triangle z$. No caso do FedAvgM, é adicionado, no servidor, um vetor de momento v (Hsu *et al.*, 2019): $z \leftarrow z - v$.

Primeiro, acontece o cálculo da média ponderada, em que o servidor calcula a média ponderada das atualizações dos participantes. Segundo, é calculado o momento no qual o servidor faz o cálculo do novo vetor de momento utilizando β e a atualização de agregação. Com isso, em terceiro momento, o modelo global é atualizado com $z \leftarrow z - v$ (Hsu *et al.*, 2019). Em que $v \leftarrow \beta v + \Delta z$.

3.3.1.3 FedProx

Em (Li *et al.*, 2020), é proposta uma técnica de agregação, FedProx, responsável por mitigar as problemáticas da variabilidade expressiva das características dos sistemas nos dispositivos e de dados distribuídos heterogeneamente. O FedProx pode ser entendido como uma generalização e reparametrização do FedAvg (Li *et al.*, 2020). No FedProx é adicionado

um termo proximal, que resolve o problema da heterogeneidade estatística, restringindo as atualizações locais a ficarem mais próximas ao modelo global inicial sem a necessidade de mudar o número de épocas locais (Li *et al.*, 2020). Esse método de agregação segue a seguinte equação:

$$\min_{z} h_i(z; z^t) = F_i(z) + \frac{\mu}{2}\|z - z_t\|^2 \tag{3.6}$$

Em que h_i é uma função substituta, $F_i(z)$ é a função de perda empírica local do cliente i, $\frac{\mu}{2}\|z - z_t\|^2$ é o termo proximal e μ é um hipermarâmetro que atua no peso da penalidade.

Já para parte do servidor, o modelo global é atualizado da seguinte maneira:

$$z_{t+1} \leftarrow \sum_{k \in S_t} \frac{1}{I} z_{t+1}^i \tag{3.7}$$

I são os dispositivos selecionados.

Além disso, permite agregar de forma segura uma quantidade variante de trabalho local resultante de sistemas heterogêneos (Li *et al.*, 2020).

3.3.1.4 FedYogi

Assim como FedAvgM e o FedProx, o FedYogi, proposto por (Reddi *et al.*, 2021), tem como objetivo melhorar a performance dos modelos em contextos não-IID em comparação com o FedAvg base. No trabalho realizado por (Reddi *et al.*, 2021), é proposto um *framework* otimizador geral em que os clientes performam diversas épocas locais de treino utilizando o otimizador do cliente, assim minimizando a Função Perda na base de dados local. Além disso, o servidor faz a atualização global do modelo aplicando um otimizador de servidor baseado em gradiente na média das atualizações dos modelos do cliente (Reddi *et al.*, 2021). O FedYogi está inserido em uma estrutura de otimização conhecida como FedOpt, na qual se divide a otimização em dois, tanto no lado do servidor quanto no lado do cliente, e é inspirado no otimizador Yogi (Reddi *et al.*, 2021).

O FedOpt funciona da maneira descrita a seguir. Primeiramente, os clientes fazem as épocas de treino com sua própria base de dados utilizando o otimizador SGD. Em segundo lugar, o usuário, depois do treinamento, calcula e envia a diferença entre os parâmetros do seu modelo e o modelo inicial do servidor. A diferença é determinada pela equação:

$$\Delta_t^j = x_{t,L}^j - x_t \tag{3.8}$$

Onde $x\{t,L\}^j$ representa o modelo do cliente j após L épocas locais de treinamento, e x_t é o modelo global no início da rodada t.

Em terceiro lugar, após os clientes enviarem as atualizações, o servidor aplica a agregação FedYogi, que irá atualizar o modelo global (Reddi *et al.*, 2021):

$$\Delta_t = \frac{1}{|S|}\sum_{j\in S}\Delta_t^j \tag{3.9}$$

S é o conjunto de clientes na rodada.

No trabalho de (Reddi *et al.*, 2021) foram propostos três agregadores diferentes: FedAdam. FedAdagrad e FedYogi. Apesar desses três agregadores terem objetivos parecidos, o FedYogi apresentou melhor desempenho comparado aos demais na maioria dos experimentos realizados no artigo.

A agregação do FedYogi funciona acumulando a cada rodada o primeiro momento e um segundo momento adaptativo, em que o FedYogi se diferencia dos demais. O primeiro momento é onde ocorrerá o acúmulo das atualizações, suavizando, assim, as oscilações (Reddi *et al.*, 2021). Seguindo a equação:

$$q_t = \gamma_1 q_{t-1} + (1-\gamma_1)\Delta_t \tag{3.10}$$

Em que q_t é o primeiro momento e γ_1 é parâmetro de decaimento do primeiro momento.

No segundo momento adaptativo, é armazenado o histórico dos quadrados das atualizações; isso permite que sejam atualizadas as taxas de aprendizado adaptativo por coordenada (Reddi *et al.*, 2021).

$$d_t = d_{t-1} - (1-\gamma_2)\Delta_t^2 \operatorname{sign}(d_{t-1} - \Delta_t^2) \tag{3.11}$$

Sendo d_t a estimativa do segundo momento, γ_2 o parâmetro de decaimento do segundo momento, Δ_t^2 é o quadrado elemento-a-elemento e sign é a função que retorna o sinal da diferença.

E por último, para a atualização do modelo global é feito o seguinte cálculo:

$$x_{t+1} = x_t + \eta\frac{q_t}{\sqrt{d_t}+\tau} \tag{3.12}$$

Onde η é a taxa de aprendizado de servidor e τ é o parâmetro de adaptabilidade.

3.3.2 Desafios e considerações

Ainda que o FL tenha muitos benefícios, essa abordagem ainda apresenta desafios importantes que podem afetar seu desempenho. Um dos obstáculos mais notórios do FL é quando clientes apresentam dados não-IID, desbalanceados e altamente variáveis em termos de quantidade e qualidade. Nessa variação de dados entre clientes, pode apresentar baixa performance entre os modelos de cada cliente (Bhanbhro *et al.*, 2024), afetando, também, o desempenho global.

Apesar disso, o FL apresenta grandes vantagens para o cenário da agricultura, como a personalização de dados por armadilha, ou seja, cada cliente possui dados característicos, privacidade de dados dos clientes e eficiência de comunicação, por não haver uma grande troca de informações entre cliente e servidor.

3.4 Contêineres

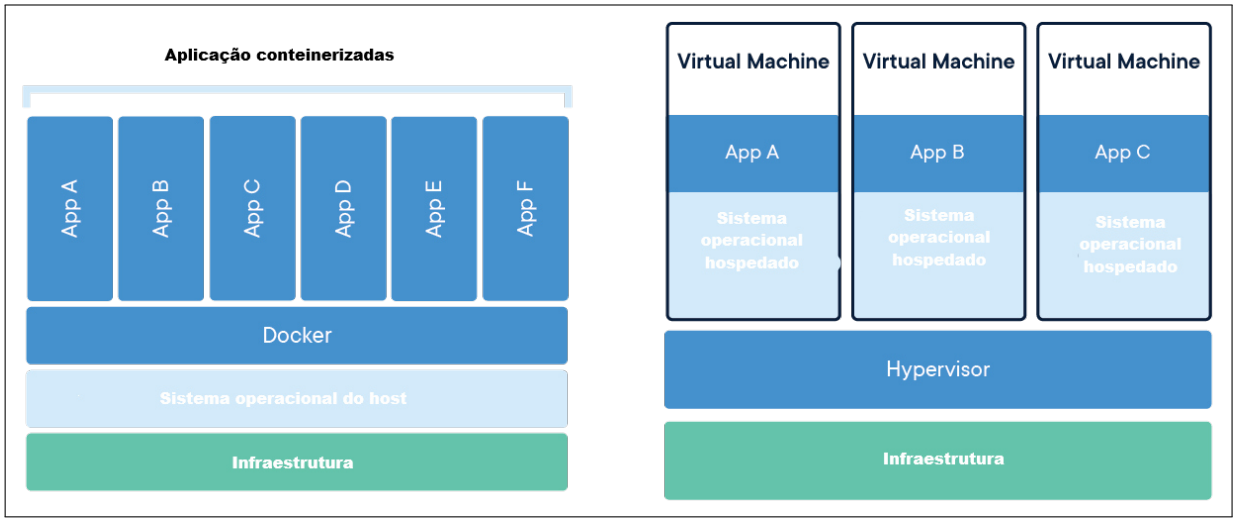
Os contêineres são pacotes de *software* que agrupam códigos e todas as suas dependências, possibilitando que a aplicação seja executada de maneira rápida e consistente em diferentes ambientes computacionais, garantindo, assim, que as aplicações funcionem uniformemente apesar das diferenças de desenvolvimento. Além disso, podem rodar múltiplos contêineres em uma mesma máquina e podem compartilhar o *kernel* do sistema operacional com outros contêineres, mas mantendo a independência de cada processo de cada contêiner (Docker, Inc., 2025b).

Os contêineres baseiam-se na conteinerização, que é uma técnica de agrupamento de códigos de uma aplicação incluindo o sistema operacional, variáveis de ambiente, bibliotecas, arquivos de configuração e demais dependências, permitindo maior portabilidade (Susnjara; SMALLEY, 2024).

Outro ponto importante é saber as principais diferenças entre contêineres e VM. As VMs são emulações ou representações virtuais de um computador físico, nas quais há um *hipervisor* que aloca recursos computacionais físicos para cada VM e cada uma possui um sistema operacional convidado e uma cópia virtual do hardware de que o sistema operacional necessita para rodar. Essa diferença é exemplificada na Figura 4 com a utilização de contêineres Docker. Diferentemente das VMs, os contêineres virtualizam o sistema operacional para que cada contêiner receba apenas a aplicação, dependências, bibliotecas e configurações, o que os

torna bem mais leves (Susnjara; SMALLEY, 2024). Além disso, as VMs necessitam de mais memória, já que cada convidado tem seu próprio sistema operacional, e precisam de um maior número de recursos (Yadav *et al.*, 2019).

Figura 4 – Comparativo Contêiners e VM



Fonte: Adaptado de (Docker, Inc., 2025b).

3.4.1 Docker

Docker é uma plataforma de virtualização baseada em contêineres, na qual se estendem funcionalidades de contêineres Linux. Possibilita um gerenciamento e uma execução flexível de contêineres entre usuários e *data centers*, sendo implementada facilmente com um arquivo Dockerfile, que especifica uma sequência de tarefas iniciais e constrói imagens personalizadas para uma aplicação de acordo com suas especificações (Bentaleb *et al.*, 2022). O Docker-Hub atua como o principal repositório de registro para compartilhar, armazenar e gerenciar imagens Docker (Docker, Inc., 2025a). Já o Docker-compose é uma ferramenta que possibilita a operação de microsserviços, em que é possível configurar a aplicação em um único arquivo YAML, em que, com apenas um comando, é possível criar e começar todos os serviços do arquivo de configuração (Bentaleb *et al.*, 2022).

4 METODOLOGIA

Neste capítulo, é detalhada a metodologia utilizada para o desenvolvimento do FL, levando em consideração a avaliação dos métodos aplicados FedAvg, FedAvgM, FedProx e FedYogi, explorando essas estratégias em um cenário de distribuição de dados não homogênea em campo de agricultura.

4.1 Base de dados

4.1.1 Descrição e origem

Neste trabalho, é utilizada uma base de dados própria do projeto de pesquisa (FarmOnEdge) da Universidade Federal do Ceará. A base de dados é composta por imagens obtidas por câmeras integradas a uma Raspberry Pi em armadilhas inteligentes (Silva *et al.*, 2024), onde é registrada a presença da praga *S. frugiperda* adulta, como mostra a Figura 5. Foram capturadas imagens no período de 28/11/2023 até 30/11/2023, nas quais foram capturados cerca de 5000 registros, que foram processados e analisados. A aquisição é descrita em mais detalhes em (Soares *et al.*, 2025).

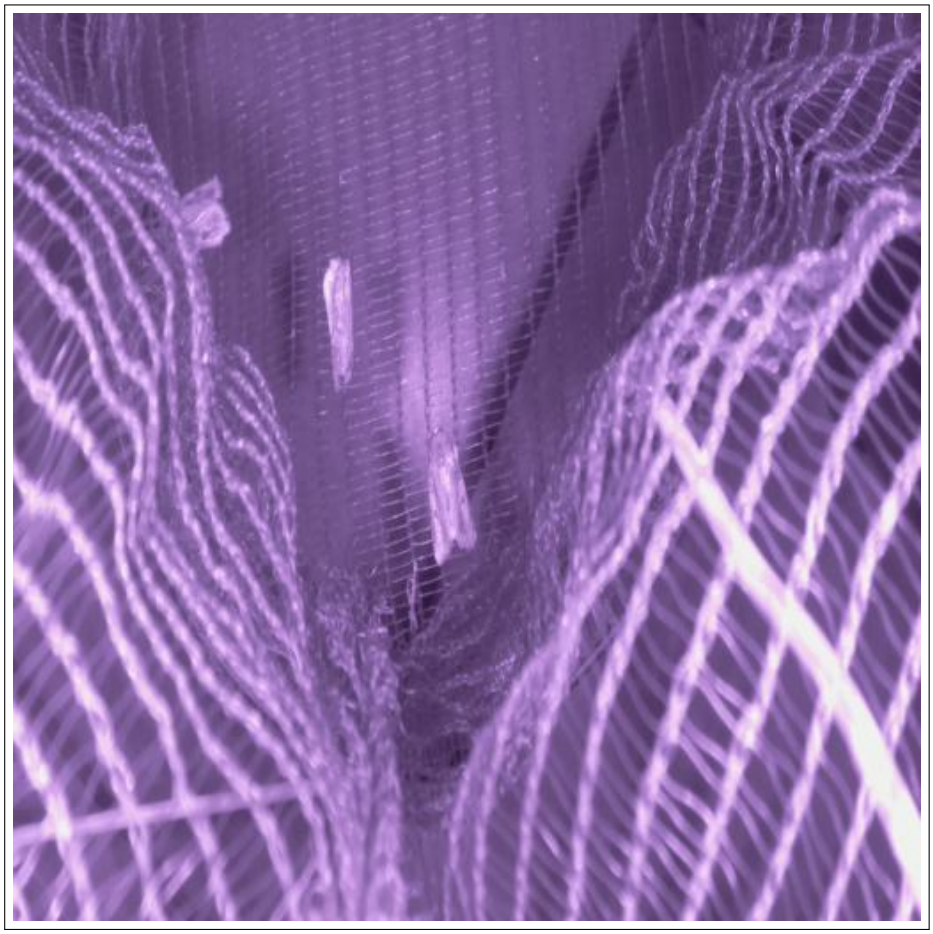
4.1.2 Pré-processamento

O pré-processamento foi realizado de acordo com (Soares *et al.*, 2025), em que as imagens obtidas foram normalizadas entre [0, 1]. Assim, mantendo uma padronização das imagens, essencial para garantir consistência. Além da normalização, as fotos também foram redimensionadas para 224x224, tamanho compatível com o modelo utilizado, MobileNet, e que utiliza menos recursos computacionais, como memória, o que é importante para o objetivo desse trabalho. Além disso, o conjunto de dados com as imagens foi dividido em dois grupos: treino e teste, o primeiro com 80% do total de imagens e o teste com o restante das imagens, cerca de 20%. O pré-processamento foi replicado para garantir a coerência e possibilitar a comparação com o método de treinamento centralizado realizado em (Soares *et al.*, 2025).

4.2 Modelos utilizados

Os modelos utilizados no trabalho foram escolhidos de acordo com o (Soares *et al.*, 2025), no qual foi realizado um projeto de classificação da *Spodoptera frugiperda* em campo

Figura 5 – Armadilha em horário noturno com a presença de insetos



Fonte: Base de dados obtida por (Silva *et al.*, 2024).

agrícola com aprendizado centralizado. A classificação realizada era binária, na qual havia duas classificações possíveis: com insetos e sem insetos. No artigo (Soares *et al.*, 2025) foram testadas seis arquiteturas de CNN pré-treinadas para a extração de características, sendo divididas em duas categorias: redes leves, como MobileNet, EfficientNetB0 e NasNetMobile, e redes densas, a exemplo da DenseNet201, ResNet50 e VGG16. Ademais, para a classificação, foram utilizados quatro algoritmos, entre eles: LightGBM, Support Vector Machine, MLP e Random Forest. Então, a escolha da combinação de modelos para extração de características com algoritmos de classificação foi feita a partir da análise de métricas de desempenho, como acurácia, kappa, precisão, revocação e F1-score presentes no artigo.

No âmbito deste trabalho, levou-se em consideração, para a escolha do modelo, aquele que possuía a menor utilização de recursos computacionais, importante para o contexto de dispositivos com recursos limitados em campo de agricultura. Mais detalhes serão apresentados na subseção seguinte.

4.2.1 MobileNet + MLP

Em primeiro caso, o modelo da MobileNet em conjunto com MLP, de acordo com o artigo (Soares *et al.*, 2025), foi, juntamente com DenseNet201, um dos modelos que possuíam as melhores métricas de avaliação. Em segundo caso, para escolher entre MobileNet e DenseNet201, foi escolhido aquele que apresentava, quando embarcado, o menor custo computacional, bem como a utilização de memória.

4.2.1.1 Configuração das arquiteturas

O Quadro 2 apresenta a configuração utilizada para cada arquitetura. A MobileNet foi pré-treinada baseada na base de dados da ImageNet, foi definida como padrão de entradas de dimensões 224*x*224*x*3, sinalizado como *input_shape*. Para a operação de *pooling* foi utilizado o *GlobalAveragePooling2D*. Este método transforma padrões visuais de imagens em um vetor de características de 1024 dimensões. No MLP recebe como entrada 1024 neurônios, os quais vêm da camada de *GlobalAveragePooling2D* do MobileNet. Além disso, o MLP possui outras quatro camadas, sendo a primeira camada oculta com 200 neurônios e ativação ReLU, duas camadas com 100 neurônios cada, também com ativação ReLU, e por fim a camada de saída que recebe apenas um neurônio com ativação *Sigmoid*. O otimizador utilizado é o Adam e a função de perda é a Entropia Cruzada Binária, muito utilizada para classificações binárias. Todos esses parâmetros foram escolhidos conforme (Soares *et al.*, 2025).

Quadro 2 – Configuração das arquiteturas utilizadas neste trabalho

Arquitetura	**Tipo**	**Configuração**
MobileNet	Extrator de características	• Pré-treinada (ImageNet) • `input_shape=(224, 224, 3)` • *GlobalAveragePooling2D*
MLP	Classificador	• Input: 1024 neurônios • Dense: 200 (ReLU) • Dense: 100 (ReLU) – 2 camadas • Dense: 1 (Sigmoid) • Otimizador: Adam • Função de perda:*Binary Crossentropy*

Fonte: Elaborado pela autora (2025).

4.3 Ambiente Experimental

Para realizar uma simulação mais realista do FL em campo de agricultura, foram utilizados ferramentas e *frameworks* para a configuração dos experimentos que serão apresentados nas subseções seguintes. Todas as simulações foram realizadas em uma máquina com GPU NVIDIA GeForce RTX 3050 (4 Gigabyte (GB) RAM), CPU Intel i5-12450H e 24 GB de RAM, utilizando o sistema operacional Windows.

4.3.1 Ferramentas utilizadas

4.3.1.1 Flower

Flower é o *framework* responsável pela simulação do FL e possibilita a implementação realista de um cenário, coordenando a integração entre treinamento de cliente e agregação do servidor. Além disso, *Flower* oferece a possibilidade de uma transição rápida de *pipelines* de treinamento de modelos de ML já existentes para um cenário federado. Ademais, esse *framework* oferece suporte para diversas simulações de ambientes federados (Beutel *et al.*, 2022), por exemplo, a utilização de aplicações *mobile* integradas com FL.

4.3.1.2 Tensorflow

Tensorflow é uma biblioteca para ML utilizada para o desenvolvimento e treinamento dos modelos de DL (TensorFlow, 2025), ML em larga escala e outras tarefas de análise preditiva e estatística (Databricks, Inc., 2025). Essa biblioteca permite a implementação de modelos de ML de forma mais fácil e rápida, agilizando o processo (Databricks, Inc., 2025).

4.3.1.3 Docker

Docker foi utilizado para simular um cenário mais realista do FL, no qual há a separação dos ambientes de execução de cada cliente. Além de auxiliar na análise de recursos computacionais utilizados por cada contêiner (servidor e cliente). Ademais, é possível ter uma maior reprodutibilidade do projeto.

4.3.1.4 Configuração de recursos computacionais para simulação

Levando em consideração a armadilha inteligente proposta no artigo (Silva *et al.*, 2024), em que se utiliza uma câmera integrada a uma *Raspberry Pi* 4, foram analisadas as configurações mínimas de *hardware* para esse dispositivo de acordo com a documentação oficial, o qual pode possuir de 1 a 8 GB de memória RAM e 4 núcleos de CPU. Portanto, para cada cliente (armadilha inteligente) foi atribuída uma limitação de 1,5 GB RAM e 1 núcleo de CPU. A restrição de memória também levou em consideração os limites computacionais da máquina utilizada para a realização dos experimentos, não podendo exceder 1,5 GB de memória por cliente. Além disso, a quantidade de núcleos de CPU e a memória atribuída a cada cliente foram reduzidas em função das limitações da máquina na qual os experimentos foram conduzidos.

4.4 Configuração para Aprendizado Federado

Nesta pesquisa, foram escolhidos os seguintes critérios para a configuração do FL:

1. Número de rodadas: 200 globais

   Inicialmente, foram testadas as agregações com cem rodadas para analisar a convergência das agregações e como se comportavam. Após isso, foi percebido que cem rodadas não eram suficientes para analisar o comportamento de uma forma geral. O número 200 foi escolhido conforme o artigo (Li *et al.*, 2020), pois corresponde ao maior número de rodadas, entre os trabalhos de cada agregação, que possibilitou a execução das agregações no ambiente computacional disponível, sem causar gargalos ou tempo de execução excessivo.
2. Épocas locais: 3

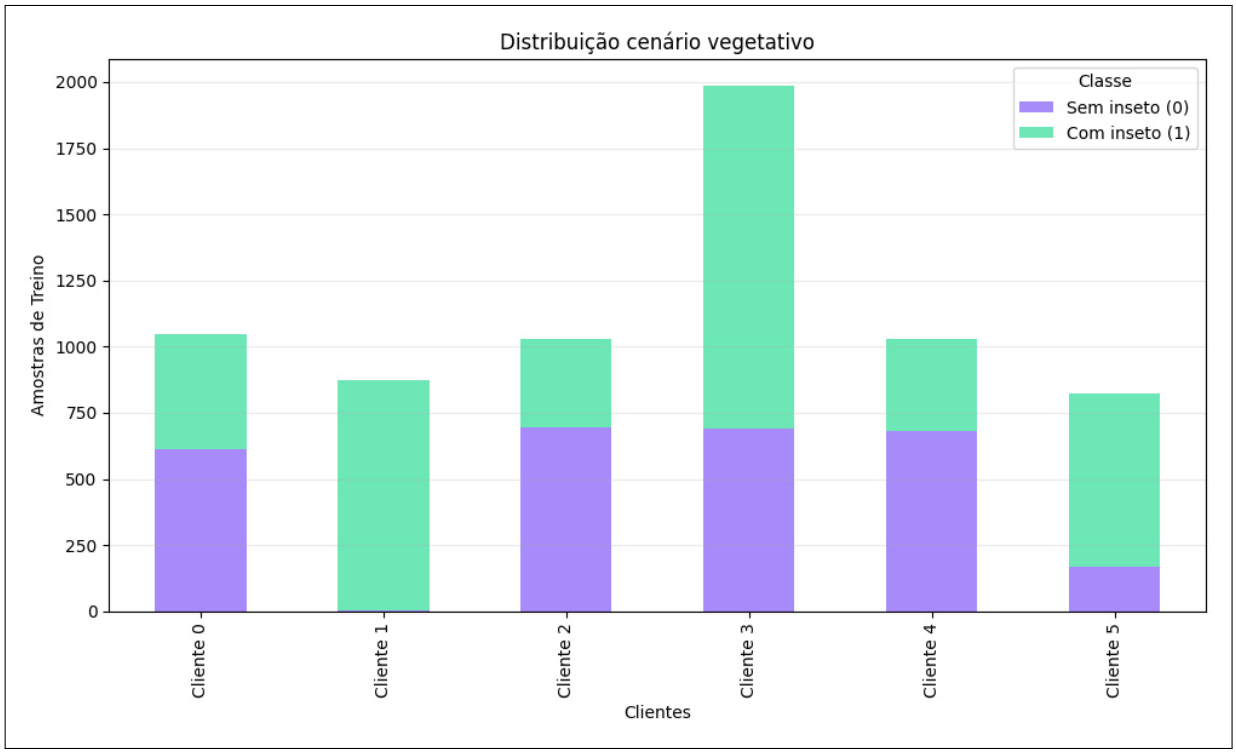
   Como a maioria dos trabalhos presentes na literatura usa como padrão épocas de 1 a 5, foram testados inicialmente com 1, 3 e 5. O valor de 3 épocas foi o que obteve o melhor resultado tanto em relação a métricas quanto em relação ao tempo de simulação.
3. *Batch*: 32

   O número de *batchs* utilizado foi selecionado de acordo com o experimentado no artigo (Soares *et al.*, 2025) para manter consistência entre os experimentos. Foram testados também com 16 *batchs*, porém, levando em consideração acurácia e F1-score, os experimentos com 32 obtiveram melhores resultados.
4. Distribuição de dados: Não-IID

   Neste trabalho, um dos objetivos principais é a simulação mais próxima ao real de uma

armadilha inteligente em um campo agrícola. Essa simulação representa tanto a reprodução mais realista da disposição de armadilhas inteligentes em campo de cultivo quanto a distribuição da praga nas diferentes fases do milho. Logo, a distribuição foi feita em dois contextos diferentes: vegetativo e reprodutivo. Na época vegetativa do milho há uma incidência maior de insetos em relação à época reprodutiva da planta, como é mencionado na tese (Praxedes Junior, 2024). Segundo essa tese, na fase vegetativa há cerca de 3 vezes mais ocorrência da praga em relação à fase reprodutiva. Então, para a divisão dos dados foi levada em consideração (Praxedes Junior, 2024) e está apresentada nas Figuras 6 e 7. Ademais, na divisão dos dados, há diferença na quantidade de imagens entre os cenários. Além disso, nas Figuras 6 e 7, são apresentados 6 clientes, de 0 a 5, que possuem quantidades diferentes de dados e de classes. Cada cor da barra representa uma classe e a quantidade de insetos por cliente.

Figura 6 – Gráfico de distribuição para cenário vegetativo



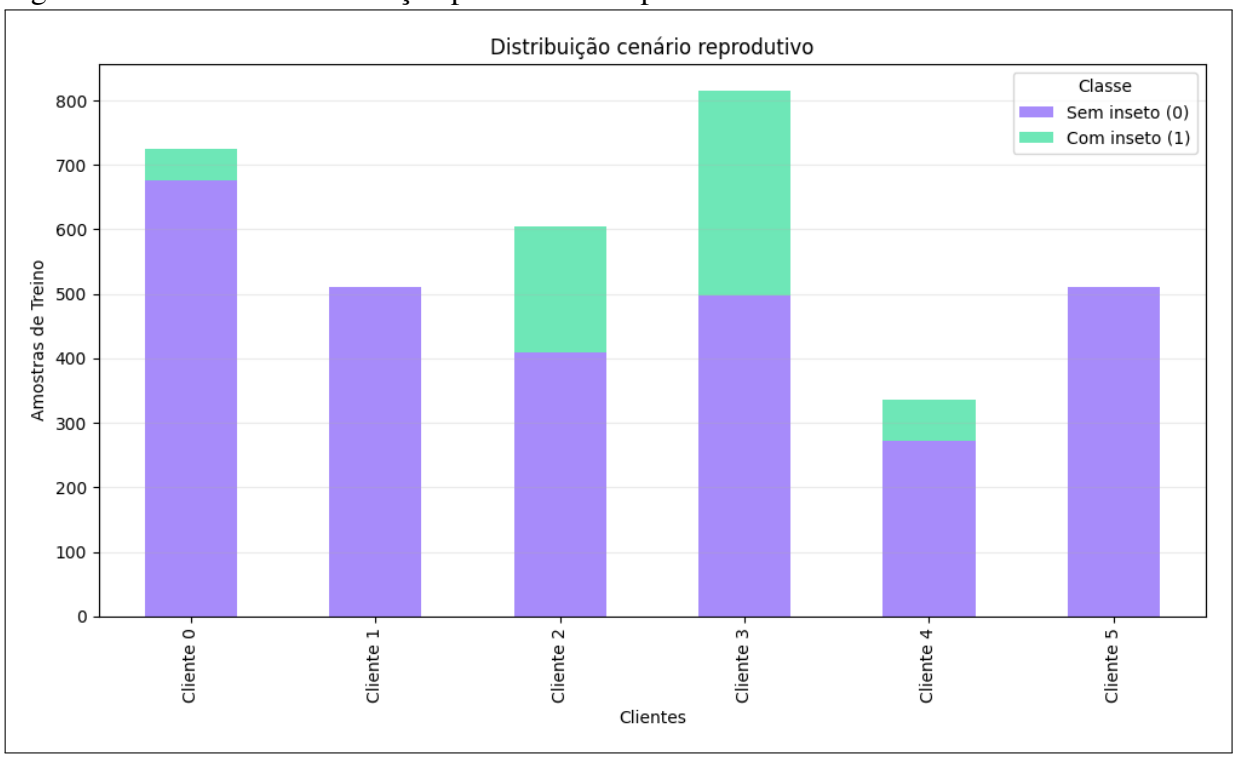
Fonte: elaborado pela autora (2025).

5. Número de Clientes: 6

   O número de clientes foi baseado na tese (Praxedes Junior, 2024), na qual o autor propôs a distribuição de 6 armadilhas inteligentes em um campo de agricultura para a classificação da *Spodoptera frugiperda* adulta, com uma área de produção agrícola de 60 hectares.

6. Comunicação e avaliação

Figura 7 – Gráfico de distribuição para cenário reprodutivo



Fonte: elaborado pela autora (2025).

Durante as rodadas, todos os clientes participaram e todos também foram avaliados. Essa condição foi estabelecida, principalmente por se tratar de uma simulação com poucos clientes, sendo de suma importância a participação e a avaliação de todos para a obtenção de um melhor resultado.

As informações da configuração do treinamento federado foram sintetizadas no Quadro 3.

Quadro 3 – Configuração do treinamento federado

Parâmetro	**Descrição**
Número de rodadas	200 rodadas globais
Número de Épocas locais	3 épocas
Número de *Batch's*	32
Distribuição dos dados	Não-IID
Número de clientes	6 clientes participantes
Comunicação e avaliação	Participação e avaliação de 100% dos clientes

Fonte: Elaborado pela autora (2025).

4.5 Estratégias de Agregação

Nessa seção são apresentadas as agregações utilizadas para avaliar o comportamento do FL em diferentes níveis de dados não homogêneos.

Proposto por (McMahan *et al.*, 2017), o *FederatedAveraging*, ou FedAvg, é um dos métodos de agregação mais utilizados no contexto do FL. Esse método foi implementado servindo como base, já que é uma agregação fundamental do FL. Com o *framework* Flower, essa estratégia já vem implementada de forma padrão.

O FedAvgM foi implementado utilizando o *template* base do *framework* Flower em: <https://github.com/adap/flower/tree/main/baselines/fedavgm>, e o artigo (Hsu *et al.*, 2019). A escolha de parâmetros foi feita a partir do artigo (Hsu *et al.*, 2019). Foram testados β iguais a 0,7, 0,9 e 0,99. O valor de β igual a 0,9 teve o melhor resultado.

Por ser uma estratégia base e fundamental em relação a dados não-IID, é de suma importância a implementação dessa técnica para os cenários propostos nesse trabalho e, assim, analisar sua eficácia nessa pesquisa. O FedProx foi implementado a partir de um exemplo de algoritmo presente no GitHub do *framework* Flower: <https://github.com/adap/flower/tree/main/baselines/fedprox>, e do algoritmo demonstrado em (Li *et al.*, 2020). Para a escolha do parâmetro do μ, foram testados valores de 1.0 e 0.01, valores que, no artigo (Li *et al.*, 2020), foram descritos como valores que conseguiram alcançar mais estabilização e menor *loss* para os conjuntos de dados utilizados.

FedYogi, assim como as outras agregações, foi escolhido por sua eficiência demonstrada no artigo em que foi proposto (Reddi *et al.*, 2021), em obter melhores resultados em relação a outras estratégias de agregação, como FedAvgM, SCAFFOLD, o clássico FedAvg. O FedYogi foi implementado com base no artigo (Reddi *et al.*, 2021) e em repositórios presentes na plataforma GitHub: <https://github.com/BalajiAI/Federated-Learning>. Os valores de gama foram escolhidos de acordo com o padrão do artigo, nos valores de 0,0 e 0,99.

Também foi testada a agregação FedAdam; contudo, FedYogi obteve melhor desempenho em relação às métricas avaliadas.

4.6 Métricas de Avaliação

Para a avaliação dos métodos implementados, foi determinada a utilização de métricas tanto de avaliação para classificação do modelo quanto de desempenho computacional.

Essas métricas são importantes para analisar a capacidade de um método ser aplicado em um cenário realista.

4.6.1 Métricas de avaliação do modelo

Para avaliar a classificação feita pelo aprendizado foram utilizadas seis métricas:

A acurácia é a taxa que mede de todas as classificações realizadas, quais foram classificadas corretamente, e calculada pela equação:

$$A = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

Definem-se TP como verdadeiro positivo: quando o modelo classifica corretamente uma classe como positiva, que realmente é positiva; TN como verdadeiro negativo: quando o modelo classifica corretamente uma classe como negativa, que realmente é negativa; FP como falso positivo: quando o modelo classifica incorretamente uma classe como positiva, mas que realmente é negativa; FN como falso negativo: quando o modelo classifica incorretamente uma classe como negativa, mas que realmente é positiva.

Precisão calcula a taxa de classificações positivas realizadas pelo modelo e que são realmente positivas. É obtida por:

$$P = \frac{TP}{TP + FP} \tag{4.2}$$

Recall, ou também conhecido como taxa de verdadeiro positivo, calcula a proporção de todos os positivos reais os quais foram classificados corretamente. É dado por:

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

F1-score pode ser interpretado como a média harmônica da precisão e *recall*. É calculada pela seguinte equação:

$$F1 = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{4.4}$$

Kappa é uma métrica estatística, na qual avalia a concordância entre classificações e é obtida a partir do cálculo:

$$Kappa = \frac{(p_0 - p_e)}{(1 - p_e)} \tag{4.5}$$

Onde p_0 é a proporção de acertos que foram observados e p_e é a proporção de acertos esperados pelo acaso.

A função perda, conhecida como *loss*, é uma métrica que calcula o desvio das previsões de um modelo em relação às previsões corretas. Além disso, neste trabalho foi utilizada a função perda Entropia Cruzada Binária, adequadas para tarefas de classificação binária.

4.6.2 Métricas de recursos computacionais

Para avaliar a utilização de recursos computacionais de cada cliente, foram utilizadas duas métricas, que também correspondem a recursos que foram previamente limitados. São elas: a utilização de memória RAM em Megabyte (MB) e o número de núcleos de CPU por cliente. Essas métricas são avaliadas durante o treinamento no Docker.

4.7 Procedimento

O procedimento metodológico deste trabalho foi dividido em cinco etapas principais: (i) definição da base de dados e modelo, (ii) aplicação do FL, (iii) simulação de cenário real com limitação de recursos computacionais, (iv) aplicação de agregações, (v) teste e análise de resultados.

Na primeira etapa (definição da base de dados e modelo), a base de dados foi obtida a partir do artigo (Silva *et al.*, 2024), onde foram capturadas imagens da *S. frugiperda* em uma armadilha inteligente e montada a base de dados com mais de cinco mil imagens. Os modelos testados foram definidos pelo artigo (Soares *et al.*, 2025), no qual foi utilizada a mesma base de dados, e foram testados seis modelos no total. Neste trabalho foram escolhidos os modelos que tiveram os melhores resultados em relação às métricas do artigo citado.

Na segunda etapa (aplicação do FL), foi definido o *framework* Flower para a implementação do projeto. O Flower oferece *templates* base para a simulação e agregações para reproduzir no FL personalizado.

Na terceira fase (simulação de cenário real com limitação de recursos computacionais) foram pesquisados estudos e artigos demonstrando a distribuição de armadilhas inteligentes

em campo de agricultura e a distribuição de insetos em épocas do ano. No trabalho de (Praxedes Junior, 2024), apresentava a distribuição de seis armadilhas em um campo. A quantidade de armadilhas foi escolhida para ser o número de clientes. Além disso, nesse artigo, relatava que na época vegetativa do milho era cerca de três vezes maior a incidência de insetos em relação à época reprodutiva. Esses cenários, reprodutivo e vegetativo, foram replicados na simulação do FL, em que foi dividida a base de dados de acordo com a incidência de insetos em cada fase do milho. A limitação de recursos foi realizada com Docker, onde foram definidos 1 núcleo de CPU e 1.5 GB RAM de memória para cada cliente.

A quarta fase (aplicação de agregações) foi implementada a partir de exemplos presentes no repositório GitHub do *framework* Flower. Foram aplicadas três agregações além de FedAvg, que serviu de *baseline* para outras três agregações FedProx, FedAvgM e FedYogi, que têm como objetivo principal lidar com dados não-IID. Todas as agregações foram executadas com o mesmo número de rodadas e as mesmas distribuições de dados para ser possível a comparação mais justa de desempenho entre elas. Foram levados em consideração o desempenho do modelo global, assim como a utilização média de recursos computacionais por cliente em cada agregação. Em alguns métodos houve a escolha de parâmetros como o termo proximal do FedProx e o *momento* (β) de FedAvgM. Esses parâmetros foram obtidos por meio da experimentação, em que foram selecionados aqueles que tiveram melhor desempenho nas métricas em relação a outros valores. Todas as agregações foram executadas em ambiente Docker com limitação de recursos. A implementação de código desenvolvida neste trabalho está disponibilizada no repositório GitHub, em: <https://github.com/gabisponciano/farmonedge-FL> .

A quinta e última etapa (teste e análise de resultados) foi realizada utilizando todas as ferramentas citadas anteriormente, comparando os cenários vegetativo e reprodutivo e os resultados obtidos, respectivamente.

5 ANÁLISE DE RESULTADOS

Neste capítulo, foram analisados os resultados obtidos a partir dos experimentos realizados com a implementação de métodos apresentados na literatura que apresentam como objetivo a redução da problemática de dados não-IID. Para essa análise, foram simulados dois cenários de distribuição não homogênea. Em cada cenário, determinados clientes possuem mais dados de uma classe específica, de acordo com a época do milho: vegetativa ou reprodutiva.

5.1 Cenário Reprodutivo

5.1.1 Métricas de avaliação

Na Tabela 1 podemos observar quatro métricas de agregação no cenário reprodutivo. A agregação FedProx, utilizando o parâmetro $\mu = 0.01$, obteve o melhor resultado entre as quatro agregações testadas. A segunda agregação que apresenta o melhor desempenho é a FedYogi; porém, seus resultados não apresentam melhorias significativas em relação à FedAvgM. A FedAvg foi a agregação que obteve o pior desempenho em três métricas, obtendo resultado superior a FedAvgM apenas em função perda, com uma diferença de aproximadamente 0,18.

Tabela 1 – Métricas globais de desempenho dos agregadores - Reprodutivo

Agregador	Acurácia	F1-score	Kappa	Função Perda
FedAvg	0.795	0.786	0.603	1.256
FedAvgM	0.814	0.818	0.645	1.440
FedProx	**0.879**	**0.895**	**0.753**	**0.701**
FedYogi	0.820	0.821	0.648	1.479

Fonte: Elaborado pela autora (2025).

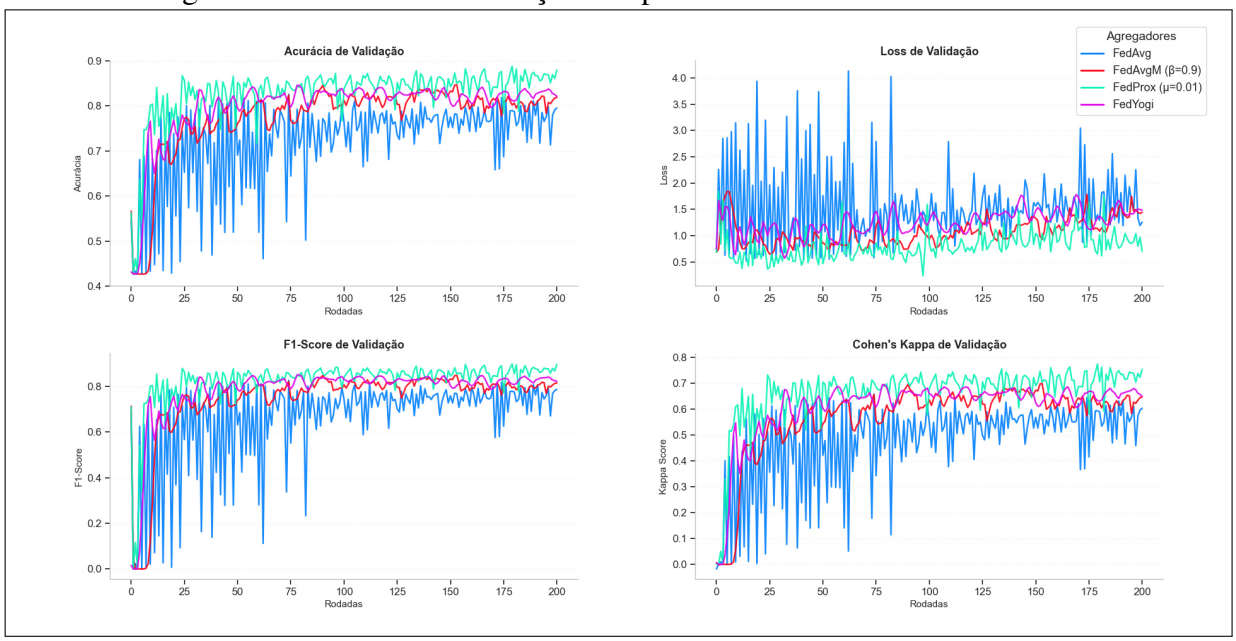
Na Figura 8 são representadas as métricas de avaliação para cada método de agregação durante 200 rodadas globais de treinamento. O primeiro gráfico apresenta a acurácia de validação, o segundo a função de perda de validação, o terceiro o F1-score de validação e, por fim, o Kappa de validação. Além disso, a Figura 8 mostra os desempenhos das agregações em relação a dados não-IID ao longo das rodadas. Os resultados evidenciam o desempenho superior do FedProx em relação às demais agregações, apresentando o melhor resultado. Esse comportamento pode ser justificado pelo termo proximal adicionado ao FedProx, que penaliza grandes divergências em relação ao modelo global, mantendo valores mais estáveis durante o treinamento.

Já em relação ao desempenho de FedAvgM e FedYogi, observa-se comportamento semelhante tanto em estabilidade quanto em resultados. No caso do FedAvgM, a adição do *momentum* no servidor resultou em melhorias nos resultados dessa agregação em relação ao FedAvg. Já o FedYogi, ao aplicar o otimizador adaptativo no servidor, demonstrou que os resultados obtiveram melhorias na maioria das métricas, com exceção da função perda. Contudo, os ajustes não foram suficientes para obter resultados em destaque nesse cenário, possivelmente pela escolha de hiperparâmetros não ideal.

FedAvg apresenta grande instabilidade ao longo das rodadas, com maior estabilidade nas últimas 25 rodadas. Isso pode ser explicado pelo fato de FedAvg não ser robusto em relação a dados não-IID, já que o FedAvg realiza apenas a média ponderada das atualizações locais, o que não leva em consideração a divergência dos clientes, típica em cenários com dados distribuídos heterogeneamente.

A maioria das agregações apresentou convergências em momentos semelhantes, por volta da rodada 70. A instabilidade observada nas rodadas é típica de cenários em que a distribuição de dados é heterogênea.

Figura 8 – Métricas de avaliação - Reprodutivo



Fonte: Elaborado pela autora (2025).

A melhor rodada para FedAvg foi a 74, em que houve acurácia de 0,825 e F1-score de 0,826, porém, as rodadas apresentam grande irregularidade. A FedAvgM obteve melhor resultado na rodada 153 com acurácia de 0,846 e com F1-score de 0,848. Após essa rodada, há uma pequena piora no desempenho. Já o FedProx obteve na rodada 179 uma acurácia de

0,887 e F1-score de 0,897, mantendo-se com um bom resultado até a rodada 200. FedYogi teve seu melhor resultado em 79, com acurácia no valor de 0,845 e F1-score de 0,850, mantendo-se relativamente estável após essa rodada, com apenas algumas irregularidades.

5.1.2 Métricas Computacionais

As métricas computacionais do cenário reprodutivo foram medidas a cada 10 segundos durante toda a execução de rodadas de cada agregação. As Tabelas 2 e 3 mostram média de utilização de CPU em porcentagem e de memória em *megabytes* dos seis clientes participantes das rodadas.

Tabela 2 – Uso de CPU por agregador - Reprodutivo

Agregador	Média (%)	Mínimo (%)	Máximo (%)
FedAvg	64.865	0.850	105.770
FedAvgM	66.432	**0.670**	**104.160**
FedProx	**56.218**	**0.670**	109.260
FedYogi	69.132	0.760	105.730

Fonte: Elaborado pela autora (2025).

Tabela 3 – Uso de Memória por agregador - Reprodutivo

Agregador	Média (MB)	Mínimo (MB)	Máximo (MB)
FedAvg	798.720	**352.100**	1247.232
FedAvgM	**786.285**	358.900	1214.464
FedProx	1162.778	352.600	1536.000
FedYogi	793.188	436.100	**1082.368**

Fonte: Elaborado pela autora (2025).

Percebe-se que, em exceção ao FedProx, que obteve uma média de utilização de 56,218% da CPU, as outras agregações tiveram valores parecidos em torno de 65%–69%. Já nos mínimos, FedAvgM e FedProx tiveram os menores valores, essa utilização mínima da CPU, geralmente ocorre quando os clientes não estão ativamente realizando os treinamentos. Todas as agregações excederam o limite da utilização de um núcleo de CPU, em até 10%, no caso do FedProx. Isso ocorre principalmente quando os usuários estão realizando os treinos.

A agregação que mais utilizou memória foi FedProx utilizando uma média de 1162,778 MB, um valor muito mais elevado do que as outras agregações testadas, que ficaram próximas em torno de 790 MB. Esse comportamento do FedProx pode acontecer por conta do cálculo do termo proximal utilizando mais memória ao realizar as rodadas. No máximo de memória utilizada, FedProx foi a única agregação a extrapolar o limite definido de 1,5 GB. Já as

outras agregações conseguiram se manter na faixa restringida. As agregações FedAvg, FedAvgM e FedYogi obtiveram valores médios próximos e inferiores ao FedProx. O comportamento do FedAvg pode ser explicado por apresentar maior simplicidade em sua implementação, apresentando um consumo de memória reduzido. O FedAvgM, apesar de adicionar o *momentum* no cálculo da agregação, não apresentou diferença significativa no consumo de memória.

O FedYogi, além de apresentar médias parecidas no consumo de memória, também obteve menor consumo máximo, o que evidencia uma maior estabilidade.

5.2 Cenário Vegetativo

5.2.1 Métricas de avaliação

No cenário vegetativo é, também, caracterizado pela maior quantidade de imagens, o que pode explicar os desempenhos das métricas da Tabela 4, que ficaram mais próximas em relação ao cenário reprodutivo.

Tabela 4 – Métricas globais de desempenho dos agregadores - Vegetativo

Agregador	Acurácia	F1-score	Kappa	Função Perda
FedAvg	0.821	0.861	0.617	1.046
FedAvgM	0.830	0.881	0.637	0.827
FedProx	**0.871**	**0.896**	**0.730**	0.695
FedYogi	0.857	0.887	0.697	**0.582**

Fonte: Elaborado pela autora (2025).

A Tabela 4 mostra que FedProx conseguiu superar em três métricas, acurácia, F1-score e kappa, os outros três algoritmos, mostrando-se robusto em manter desempenho apesar da heterogeneidade da distribuição. A segunda estratégia com melhor desempenho foi a FedYogi que se mostrou robusta e eficiente, obtendo bons valores de acurácia, f1-score, kappa e o menor valor de função perda entre as agregações. Como esperado, o FedAvg apresentou o pior desempenho.

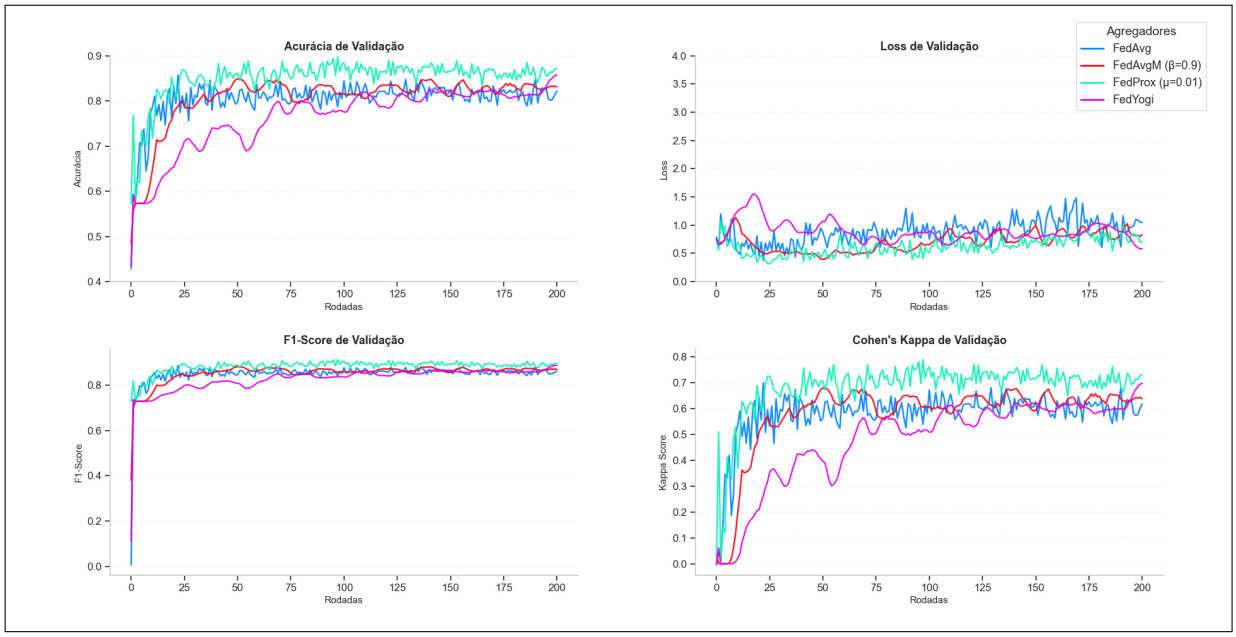
Em relação a melhores resultados, FedAvg alcançou acurácia de 0,857 e f1-score de 0,885 na rodada 22. Após essa rodada, manteve-se com valores no intervalo de diferença de 0,4. A agregação FedAvgM obteve melhores resultados na rodada cinquenta, com acurácia no valor de 0,849 e F1-score de 0,881, em que manteve o bom resultado, mas houve uma queda nas métricas à medida que as rodadas foram acontecendo. FedProx na rodada noventa e sete

teve acurácia de 0,897 e F1-score de 0,912 e manteve bons resultados. Já no FedYogi, teve seu maior desempenho na rodada final, com acurácia em 0,857 e F1-score em 0,887; se manteve bem estável nas últimas cinco rodadas.

Nos gráficos da Figura 9, é notório que as curvas das métricas são semelhantes. O gráfico sustenta os resultados da Tabela 4, na qual FedProx apresentou o melhor desempenho em três métricas. Além disso, é perceptível que FedYogi levou mais rodadas para atingir valores próximos aos dos demais algoritmos.

Os resultados obtidos dos métodos de agregação podem ser justificados com os argumentos do cenário reprodutivo. Contudo, todas as métricas obtiveram, em geral, resultados melhores. Esse comportamento pode ser explicado pelo fato de que no cenário vegetativo houve uma maior quantidade de imagens, resultando em melhores resultados. Além disso, o FedYogi, nesse cenário, obteve resultados com maior destaque. Esse desempenho pode estar relacionado à escolha de hiperparâmetros mais adequadamente otimizadas para o cenário vegetativo.

Figura 9 – Métricas de avaliação - Vegetativo



Fonte: Elaborado pela autora (2025).

5.2.2 Métricas computacionais

Assim como no cenário reprodutivo, as agregações têm o mesmo comportamento. FedProx teve a menor média de utilização de CPU, seguida da FedYogi. Na questão da utilização máxima, todas excederam o limite em mais de 10%, mostrado na Tabela 5.

Tabela 5 – Uso de CPU por agregador - Vegetativo

Agregador	Média (%)	Mínimo (%)	Máximo (%)
FedAvg	63.342	0.770	112.860
FedAvgM	61.496	0.720	**111.660**
FedProx	**32.349**	**0.700**	111.920
FedYogi	54.390	0.750	112.900

Fonte: Elaborado pela autora (2025).

A Tabela 6 mostra que a média de consumo de memória foi maior em FedProx, pelo mesmo motivo do cenário reprodutivo. Já o uso máximo de memória, todas as estratégias ultrapassaram o limite de 1.5 GB.

Diferentemente do contexto reprodutivo, o cenário vegetativo possui um maior consumo de memória em todas as agregações, já que este cenário possui mais imagens distribuídas por cliente; logo, foi necessária a utilização de mais memória.

Além disso, o FedProx apresentou o menor valor mínimo no uso de memória, que pode estar associado a períodos mais extensos sem a execução de rodadas durante o tempo total de treinamento.

Tabela 6 – Uso de Memória por agregador - Vegetativo

Agregador	Média (MB)	Mínimo (MB)	Máximo (MB)
FedAvg	**958.832**	354.400	1534.976
FedAvgM	984.252	541.800	1534.996
FedProx	1390.197	**76.210**	1536.000
FedYogi	1027.444	617.000	**1516.000**

Fonte: Elaborado pela autora (2025).

5.3 Discussão dos resultados

O objetivo desse trabalho é analisar as agregações que possuem o melhor desempenho quando simulados dados não-IID, considerando o cenário realista e heterogêneo do setor agrícola. Em síntese, foi notável que os métodos de agregação propostos na literatura para resolver a problemática da distribuição de dados não homogêneos conseguiram superar, pelo menos em um dos cenários, as métricas de avaliação do método de agregação base FedAvg. Contudo, é preciso uma análise mais precisa dos resultados obtidos.

5.3.1 Treinamento Centralizado

Esse trabalho foi baseado na pesquisa (Soares *et al.*, 2025), na qual foi realizada a classificação da *S. frugiperda* utilizando como extrator de características a Mobilenet e como modelo para classificação o MLP. No treino centralizado, conforme apresentado no artigo (Soares *et al.*, 2025), foram alcançados os seguintes resultados: acurácia de 0,962, kappa de 0,861 e f1-score de 0,9401. Esses valores representam desempenho superior em relação aos resultados nos cenários simulados com FL. Nesses cenários, o FedProx apresentou os melhores resultados. No cenário reprodutivo, o método alcançou acurácia de 0,879, kappa de 0,753 e f1-score de 0,895. Já na simulação vegetativa foram alcançados valores para acurácia de 0,871, kappa de 0,730 e f1-score de 0,896. Esses resultados são esperados, principalmente em um cenário não-IID desbalanceado, já que a heterogeneidade da distribuição eleva a complexidade do processo de aprendizado. Apesar disso, os resultados obtidos com as agregações foram satisfatórios, mostrando desempenho promissor, já que conseguiram alcançar métricas adequadas para o contexto do FL e possuem os benefícios desse aprendizado, como a maior privacidade dos dados individuais e eficiência de comunicação, fundamental para o cenário agrícola.

5.3.2 Cenários e agregações

No cenário reprodutivo, é possível perceber que o método de agregação FedProx apresentou o melhor desempenho entre todas as outras agregações, com métricas satisfatórias em dados não-IID, por apresentar bons valores em comparação com resultados presentes na literatura para dados heterogêneos. Essa agregação apresentou um consumo de memória consideravelmente maior em relação às outras métricas. Essa extrapolação do limite indica que em um cenário real, seriam imprescindíveis armadilhas inteligentes com uma maior quantidade de memória, isso é possível com a *Raspberry Pi 4* que possui modelos de até 8 GB. As outras agregações, com exceção de FedAvg que apresentou muita instabilidade, apesar de possuírem métricas razoáveis, não performaram bem em função perda. Um dos principais pode ser a quantidade de rodadas insuficiente para FedYogi, em que, na literatura, são testadas de quinhentas a mil rodadas, e, para FedAvgM, a escolha de parâmetros pode não ter sido otimizada. Não foi possível reproduzir a quantidade de rodadas, já a máquina em que os experimentos foram realizados não possuía recursos computacionais necessários para executar a simulação em tempo viável.

No cenário vegetativo, assim como no reprodutivo, a estratégia que obteve o melhor

desempenho foi a FedProx que se mostrou robusta em relação aos dados não-IID. Contudo, FedYogi também apresentou bons resultados e função perda em menor valor. Além disso, FedYogi possui uma média menor no uso de memória e seu tempo de treinamento também é significativamente menor, com apenas 17 minutos para realizar as 200 rodadas. Já FedProx realizou as 200 rodadas em 64 minutos. Logo, levando em conta o custo-benefício, FedYogi se apresenta como melhor escolha, possuindo robustez e eficiência, para o contexto do campo de agricultura, no qual há uma maior limitação de recursos.

Nesses cenários não-IID, os modelos locais tendem a divergir consideravelmente do modelo global. O FedProx adiciona um termo de proximidade na função perda dos clientes, restringindo o afastamento em relação ao modelo global. Isso resulta em uma convergência mais rápida e estável, explicando seu bom desempenho nas métricas de avaliação. Apesar disso, o FedProx exige uma maior utilização de recursos computacionais, já que, a cada rodada, cada cliente deve calcular a diferença entre os pesos do modelo local e os pesos globais fornecidos pelo servidor. Além disso, o FedYogi apresenta bons resultados, em especial em cenário vegetativo, por ser uma agregação robusta a dados não-IID, utilizando otimizador Yogi, que evita divergência e promove maior estabilidade. O FedYogi também possui menor tempo de execução, que está de acordo com (Reddi *et al.*, 2021), que propõe um algoritmo de agregação sem comprometer a comunicação.

Por fim, o FedAvg apresentou resultados inferiores em comparação às demais estratégias de agregação. Isso acontece, principalmente, pois cada cliente faz apenas SGD como otimizador local, havendo maior divergência dos modelos em cenários com dados distribuídos não homogeneamente.

5.3.3 Avaliação de métricas computacionais

Como analisado e citado anteriormente, um dos objetivos desse trabalho é simular um cenário semelhante ao real usando armadilhas inteligentes que utilizam *Raspberry Pi* com câmeras integradas. Portanto, foram consideradas as especificações de *hardware* de uma *Raspberry Pi 4* para analisar a possibilidade da armadilha atuar como cliente federado.

Com os resultados obtidos, em ambos os cenários, essa possibilidade é confirmada. Em relação à CPU, considerando os cenários vegetativo e reprodutivo, a maior utilização foi de 112,9%, porém vale ressaltar que a *Raspberry Pi 4* possui em sua configuração quatro núcleos de CPU, e a taxa foi calculada baseada na utilização de apenas um núcleo. Já em relação à memória,

a maior utilização foi de 1536 MB, o que extrapolou a restrição de 1500 MB adicionada a cada cliente. Contudo, esses computadores possuem versões com até 8 GB de RAM. Dessa forma, a execução do FL é possível nesses dispositivos. Entretanto, versões com maior capacidade de memória possuem um custo econômico mais elevado, podendo chegar a 2,5 vezes mais caras que um dispositivo com menor memória disponível, tornando imprescindível uma análise mais detalhada do custo-benefício para que a solução seja economicamente viável para o agricultor.

Além disso, no cenário vegetativo houve maior utilização de CPU e memória em relação ao cenário reprodutivo, o que pode ser explicado pelo maior número de imagens nesse cenário e, assim, pela maior utilização de processamento.

5.3.4 Conclusão

Portanto, de modo geral, os algoritmos de agregação obtiveram melhores resultados na simulação vegetativa, o que está de acordo com a severidade da heterogeneidade das distribuições. O cenário reprodutivo apresenta dados mais desbalanceados e menor quantidade de imagens em relação ao cenário vegetativo.

Além disso, para uma aplicação em um cenário real, devem ser levados em consideração ambos os contextos de distribuição. FedProx demonstra boa robustez para dados não-IID em todas as conjunturas, mas possui um maior gasto de recursos computacionais, o que pode não ser interessante para o ambiente agrícola, onde os recursos computacionais são mais escassos. O FedYogi possui bons resultados nos gastos de recursos computacionais, mas não possui destaque de resultados no cenário reprodutivo, podendo ter consequências negativas para o cultivo de milho.

6 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho, foram realizadas uma análise e uma aplicação de diferentes métodos de agregação que possuem como um dos principais objetivos a redução da problemática da distribuição não homogênea de dados entre clientes. No contexto agrícola, esse desafio torna-se ainda mais evidente, já que há uma maior limitação de recursos computacionais, como CPU e memória, nos dispositivos usados, demonstrando a importância da utilização do FL. A relevância dessa pesquisa está principalmente na questão da precisão de lidar com dados não-IID em cenários agrícolas. Por isso, foram implementadas e analisadas as agregações presentes na literatura que propõem algoritmos que possuem melhores resultados em cenários heterogêneos em relação ao FedAvg básico.

Foi utilizada uma base de dados de imagens capturadas por armadilhas inteligentes que possuíam câmeras integradas a um *Raspberry Pi*. Essas imagens foram divididas em duas classes: com insetos, onde havia a presença da praga *S. frugiperda*, e sem inseto. Houve o pré-processamento dessas imagens e foi utilizado como extrator de característica o Mobilenet e como modelo classificador o MLP. A simulação do FL foi realizada utilizando o *framework* Flower em conjunto com *Tensorflow*. Além disso, foram testados quatro algoritmos de agregação diferentes, em dois cenários não-IID com limitações de recurso usando Docker, sendo eles: FedAvg, FedAvgM, FedProx e FedYogi.

No cenário reprodutivo, o FedProx obteve um melhor resultado em todas as métricas de avaliação utilizadas, como acurácia, F1-score, kappa e Função Perda, mostrando-se promissor em relação à utilização do FL em contextos não-IID. No entanto, o FedProx apresentou um maior consumo de memória em relação aos outros algoritmos de agregação, mas ainda dentro da limitação imposta para cada cliente. Já no cenário vegetativo, assim como no reprodutivo, FedProx apresentou melhores valores na acurácia, F1-score e kappa, demonstrando bons resultados, mas houve uma utilização de processamento. Além disso, FedYogi se mostrou como uma boa escolha, obtendo boas métricas computacionais e de avaliação.

A maior restrição desse trabalho foi a falta de recursos computacionais, que pode ter afetado o desempenho das agregações, principalmente no cenário vegetativo, em que era necessário um maior processamento computacional por apresentar uma maior quantidade de imagens. Além disso, outra limitação foi a falta de trabalhos e pesquisas relacionadas voltadas para o uso do FL na classificação de insetos com dados desigualmente distribuídos, para uma maior validação dos resultados obtidos. Foram realizadas buscas de artigos e pesquisas em cinco

bases de dados científicas, sendo elas: *Google Scholar*, *Scientific Electronic Library Online* (*SciELO*), *ArXiv*, *IEEE Xplore* e *ScienceDirect*.

Como trabalho futuro, destaca-se a possibilidade de novos testes em máquinas mais robustas, com maior poder de processamento e memória, viabilizando a otimização de parâmetros dos agregadores. Além disso, novos testes com uma maior quantidade de rodadas seriam imprescindíveis para analisar como as agregações se comportam, principalmente FedYogi que necessita de um maior número para obter melhor resultado. Abre-se também a oportunidade de testes com outros modelos, como a DenseNet201, para avaliar se modelos diferentes também possuem melhoras significativas no contexto desse trabalho, integrando técnicas de compressão para analisar o equilíbrio entre a eficácia do modelo global e a eficiência do uso de recursos computacionais e de rede. Outrossim, adicionar métricas relacionadas ao consumo de recursos computacionais, como calor dissipado e energia consumida durante a execução das rodadas, já que tais fatores podem ser determinantes em cenários agrícolas.

REFERÊNCIAS

Agarap, A. F. M. Deep learning using rectified linear units (ReLU). **arXiv preprint arXiv:1803.08375**, 2019. ArXiv:1803.08375v2 [cs.NE].

Ahmed, R.; Khan, F.; Khan, S.; Haji Mohd, M. N.; Mohd, H.; Waseem, A.; Khan, M. N. A.; Ali, S.; Jamal, A.; Khan, H. Federated learning-based UAVs for the diagnosis of plant diseases. In: IEEE. **2022 8th International Conference on Engineering and Emerging Technologies (ICEET)**. Kuala Lumpur, Malaysia, 2022. p. 1–6.

Amazon Web Services. **What is Reinforcement Learning?** 2024. Acesso em: 29 set. 2025. Disponível em: <https://www.aws.amazon.com/what-is/reinforcement-learning/>.

Amazon Web Services. **O que é uma rede neural?** 2025. Acesso em: 3 out. 2025. Disponível em: <https://www.aws.amazon.com/pt/what-is/neural-network/>.

Bayer Crop Science. **Corn Growth Stages and GDU Requirements**. 2025. <https://www.cropscience.bayer.us/articles/bayer/corn-growth-stages-and-gdu-requirements>. Acessado em: 2025-12-11.

Bentaleb, O.; Belloum, A. S. Z.; Sebaa, A. Containerization technologies: taxonomies, applications and challenges. **The Journal of Supercomputing**, Springer, v. 78, p. 10973–11022, 2022.

Beutel, D. J.; Topal, T.; Mathur, A.; Qiu, X.; Fernandez-Marques, J.; Gao, Y.; Sani, L.; Li, K. H.; Parcollet, T.; Gusmão, P. P. B. d.; Lane, N. D. Flower: A friendly federated learning framework. In: **Proceedings of the 2022 ACM Conference on Federated Learning**. [s.n.], 2022. Disponível em: <https://arxiv.org/abs/2007.14390>.

Bhanbhro, J.; Nisticò, S.; Palopoli, L. Issues in federated learning: some experiments and preliminary results. **Scientific Reports**, Nature Publishing Group, v. 14, n. 1, p. 29881, 2024.

Burkov, A. **The Hundred-Page Machine Learning Book**. [S.l.]: Andriy Burkov, 2019. ISBN 9781999579517.

Burkov, A. **Machine Learning Engineering**. [S.l.]: True Positive Incorporated, 2020. ISBN 9781999579579.

Databricks, Inc. **TensorFlow Guide**. 2025. <https://www.databricks.com/br/glossary/tensorflow-guide>. Acessado em: 2025-12-11.

Deng, F.; Mao, W.; Zeng, Z.; Zeng, H.; Wei, B. Multiple diseases and pests detection based on federated learning and improved faster R-CNN. **IEEE Transactions on Instrumentation and Measurement**, IEEE, v. 71, p. 1–13, 2022.

Docker, Inc. **Docker Hub**. 2025. Acesso em: 07 jan. 2026. Disponível em: <https://docs.docker.com/docker-hub/>.

Docker, Inc. **What is a Container?** 2025. Acesso em: 2 out. 2025. Disponível em: <https://www.docker.com/resources/what-container/>.

Domingos, P. A few useful things to know about machine learning. **Communications of the ACM**, Association for Computing Machinery, v. 55, n. 10, p. 78–87, 2012.

Dubey, S. R.; Singh, S. K.; Chaudhuri, B. B. Activation functions in deep learning: A comprehensive survey and benchmark. **Neurocomputing**, Elsevier, 2022. ArXiv:2109.14545v3 [cs.LG].

Ejnisman, M. W.; Battilana, C. d. C. H.; Andrade, T. B. d. O aumento do uso de tecnologia no agronegócio: uma análise sob a ótica da proteção de dados. **TECCOGS – Revista Digital de Tecnologias Cognitivas**, n. 20, p. 113–124, jul./dez. 2019. ISSN 1984-3585.

Embrapa. **Boas Práticas Agrícolas para Produção de Alimentos Seguros no Campo: Controle de Pragas**. Brasília, DF, 2005. (Série Qualidade e Segurança dos Alimentos). Programa Alimentos Seguros (PAS), Setor Campo. Convênio CNI/SENAI/SEBRAE/EMBRAPA.

Embrapa. **Milho**. 2025. Portal Embrapa — Agro em Dados. Acesso em: 2025-10-06. Disponível em: <https://www.embrapa.br/agropensa/agro-em-dados/agricultura/milho>.

Flower Labs. **What is Federated Learning?** 2025. Acesso em: 5 out. 2025. Disponível em: <https://flower.ai/docs/framework/tutorial-series-what-is-federated-learning.html>.

Gill, K. S.; Anand, V.; Chauhan, R.; Verma, G.; Gupta, R. Agricultural pests classification using deep convolutional neural networks and transfer learning. **2023 2nd International Conference on Futuristic Technologies (INCOFT)**, p. 1–6, nov. 2023.

Haykin, S. **Neural Networks and Learning Machines**. 3. ed. Upper Saddle River, New Jersey: Pearson Education, Inc., 2009. ISBN 978-0-13-147139-9.

Hsu, T.-M. H.; Qi, H.; Brown, M. Measuring the effects of non-identical data distribution for federated visual classification. **arXiv preprint arXiv:1909.06335**, 2019.

Iowa State University Extension and Outreach. **Corn Growth Stages**. 2025. <https://crops.extension.iastate.edu/encyclopedia/corn-growth-stages>. Acessado em: 2025-12-11.

James, G.; Witten, D.; Hastie, T.; Tibshirani, R.; Taylor, J. **An Introduction to Statistical Learning: with Applications in Python**. New York, NY: Springer, 2023. (Springer Texts in Statistics). ISBN 9783031380746.

Jimenez G., D.; Solans, D.; Heikkilä, M.; Vitaletti, A.; Kourtellis, N.; Anagnostopoulos, A.; Chatzigiannakis, I. Non-iid data in federated learning: A survey with taxonomy, metrics, methods, frameworks and future directions. **IEEE Communication Surveys & Tutorials**, IEEE, v. 00, n. 00, p. 1–25, September 2024. ArXiv:2411.12377v2 [cs.LG].

Karunathilake, E. M. B. M.; Le, A. T.; Heo, S.; Chung, Y. S.; Mansoor, S. The path to smart farming: Innovations and opportunities in precision agriculture. **Agriculture**, MDPI, v. 13, n. 8, p. 1593, 2023.

Li, T.; Sahu, A. K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; Smith, V. Federated optimization in heterogeneous networks. **Proceedings of Machine Learning and Systems (MLSys)**, v. 2, p. 429–450, 2020.

McMahan, H. B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B. A. y. Communication-efficient learning of deep networks from decentralized data. In: **Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)**. [S.l.]: PMLR, 2017. (Proceedings of Machine Learning Research, v. 54), p. 1273–1282.

Mendes, C. I. C.; Maranhão, J. d. S. d. A.; Bertin, P. R. B.; Mondo, V. H. V.; Pires, F. C. Governança de dados para a pesquisa agrícola: segurança jurídica e autorregulação. **Cadernos de Ciência & Tecnologia**, Embrapa, v. 40, p. e27209, may 2023. ISSN 0104-1096.

Monteiro, A.; Santos, S.; Gonçalves, P. Precision agriculture for crop and livestock farming—brief review. **Animals**, MDPI, v. 11, n. 8, p. 2345, 2021.

Oracle Corporation. **What is Machine Learning?** 2025. Acesso em: 18 set. 2025. Disponível em: <https://www.oracle.com/br/artificial-intelligence/machine-learning/what-is-machine-learning>.

O'Shea, K.; Nash, R. An introduction to convolutional neural networks. **arXiv preprint arXiv:1511.08458**, 2015. Disponível em arXiv. Acesso em: 2025-10-06. Disponível em: <https://arxiv.org/abs/1511.08458>.

Overton, K.; Maino, J. L.; Day, R.; Umina, P. A.; Bett, B.; Carnovale, D.; Ekesi, S.; Meagher, R.; Reynolds, O. L. Global crop impacts, yield losses and action thresholds for fall armyworm (*Spodoptera frugiperda*): A review. **Crop Protection**, v. 145, p. 105641, 2021. ISSN 0261-2194.

Piccialli, F.; Della Bruna, C.; CHIARO, D.; Qi, P.; Savoia, M. AGRIFOLD: AGRIculture federated learning for optimized leaf disease detection. **Expert Systems with Applications**, Elsevier, v. 289, p. 128371, 2025.

Poloni, K. **Redes neurais convolucionais: Uma breve história sobre as arquiteturas**. 2022. Publicado em: 10 mar. 2022. Acesso em: 5 out. 2025. Disponível em: <https://medium.com/itau-data/redes-neurais-convolucionais-2206a089c715>.

Prasath, B.; Akila, M. IoT-based pest detection and classification using deep features with enhanced deep learning strategies. **Engineering Applications of Artificial Intelligence**, v. 121, p. 105985, 2023. ISSN 0952-1976.

Praxedes Junior, W. **Distribuição Espaço-Temporal de *Spodoptera frugiperda* em Milho Doce para Processamento Industrial Através da Captura de Adultos**. Dissertação (Dissertação de Mestrado) — Instituto Federal Goiano – Campus Urutaí, Urutaí, Goiás, Brasil, 2024.

Ramos, H. S.; Maia, G.; Papa, G. L.; Alvim, M. S.; Loureiro, A. A. F.; Cardoso-Pereira, I.; Campos, D. H. C.; Filipakis, G.; Riquetti, G.; Chagas, E. T. C.; Barros, P. H.; Gomes, G. N.; Allende-Cid, H. Aprendizado federado aplicado à internet das coisas. In: **Livro-texto de Minicursos do XXXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos – SBRC 2021**. Brasil: SBC – Sociedade Brasileira de Computação, 2021. cap. 5, p. 196–248.

Reddi, S. J.; Charles, Z.; Zaheer, M.; Garrett, Z.; Rush, K.; Konečný, J.; Kumar, S.; McMahan, H. B. Adaptive federated optimization. In: **International Conference on Learning Representations (ICLR)**. [S.l.: s.n.], 2021.

Saif, S.; Islam, M. J.; Jahangir, M. Z. B.; Biswas, P.; Rashid, A.; Nasim, M. A. A.; Gupta, K. D. A comprehensive review on understanding the decentralized and collaborative approach in machine learning. **arXiv preprint arXiv:2503.09833**, March 2025. Submitted to arXiv on 12 Mar 2025.

Shah, D.; Gupta, R.; Patel, K.; Jariwala, D.; Kanani, J. Deep learning based pest classification in soybean crop using residual network-50. In: **IEEE 2nd International Symposium on Sustainable Energy, Signal Processing and Cyber Security (iSSSC)**. [S.l.]: IEEE, 2022. p. 1–6. ISBN 978-1-6654-9056-6.

Silva, W. d. S.; Soares, B.; Almeida, V. d. L.; Viana, L.; Pastori, P. L.; Magalhães, D. M. V.; Rocha, A. R. d. Detecção da praga *Spodoptera frugiperda* no cultivo de milho usando armadilhas inteligentes e visão computacional. In: UNIVERSIDADE FEDERAL DO CEARÁ. **Anais do Evento**. [S.l.], 2024.

Soares, B.; Silva, W.; Ponciano, G.; Stefanie, B.; Almeida, V.; Pastori, P.; Magalhães, D.; Rocha, A. Embedded neural networks for identifying *Spodoptera frugiperda* in corn plantations. 2025. Departamento de Engenharia de Teleinformática, Universidade Federal do Ceará.

Susnjara, S.; SMALLEY, I. **O que são contêineres?** 2024. Acesso em: 2 out. 2025. Disponível em: <https://www.ibm.com/br-pt/think/topics/containers>.

Tanwar, V. K.; Sarkar, S.; Singh, A. K.; Das, S. K. ReinDSplit: Reinforced dynamic split learning for pest recognition in precision agriculture. In: **Proceedings of the 21st International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)**. [S.l.]: IEEE, 2025. ArXiv:2506.13935.

TensorFlow. **TensorFlow**. 2025. <https://www.tensorflow.org/?hl=pt-br>. Acessado em: 2025-12-11.

Venkateswara, S.; Padmanabhan, J. Deep learning based agricultural pest monitoring and classification. **Scientific Reports**, Nature Publishing Group, v. 15, p. 8684, 2025.

Yadav, A. K.; Garg, M. L.; Ritika, M. Docker containers versus virtual machine-based virtualization. In: ABRAHAM, A. *et al.* (Ed.). **Emerging Technologies in Data Mining and Information Security**. [S.l.]: Springer Nature Singapore, 2019, (Advances in Intelligent Systems and Computing, v. 814). p. 141–150.