Otimização de Modelos de Aprendizado de Máquina para Detecção de Pragas via Imagens Agrícolas com Eficiência Energética e de Memória

Douglas Silva[1], **Matheus A. Esteban**[1], **Bianca Soares** [1], **Wendell Santos** [1], **Deborah Magalhães** [2], **Atslands R. Rocha** [1]

[1]Departamento de Engenharia de Teleinformática, Universidade Federal do Ceará (UFC)

[2]Universidade da Integração Internacional da Lusofonia Afro-Brasileira (Unilab – CE)

{douglascomp,matheustebanufc,biancasoares,wendellsantos}@alu.ufc.br

deborah.vm@unilab.edu.br, atslands@ufc.br

Abstract. *Optimizing machine learning models for automatic pest counting on devices with limited computational resources enables the dissemination of this technology in agricultural fields. This work investigates model compression techniques to develop a low-cost embedded pest detection system. The results indicate that the compression techniques tested have little significant impact on the performance of the YOLO model, allowing their execution with lower memory and energy consumption, without affecting the reliability of the predictions. Validation of the model on an embedded system demonstrated a 14% reduction in energy consumption while maintaining nearly constant model accuracy.*

Resumo. *A otimização de modelos de aprendizado de máquina para contagem automática de pragas através de imagens, em dispositivos com recursos limitados, viabiliza a disseminação dessa tecnologia em campos agrícolas. Neste trabalho, investigam-se técnicas de compressão de modelos para um sistema de detecção de pragas embarcado em dispositivos de baixo custo. Os resultados indicam que as técnicas de compressão avaliadas têm impacto pouco significativo na acurácia do modelo YOLO, permitindo sua execução com menor consumo de memória e energia, sem perda relevante na confiabilidade das predições. A validação do modelo em sistema embarcado demonstrou redução de 14% do consumo energético, mantendo a acurácia do modelo quase constante.*

1. Introdução

A intensificação da produção agrícola e a crescente demanda por alimentos têm impulsionado a adoção de práticas mais eficientes e sustentáveis no campo. Diante desse cenário, o manejo adequado das lavouras torna-se essencial para reduzir perdas, minimizar custos de produção e diminuir impactos ambientais associados ao uso excessivo de defensivos agrícolas [Ediagbonya et al. 2025]. Além disso, a variabilidade das condições climáticas e a rápida proliferação de pragas exigem métodos de acompanhamento contínuos e precisos, capazes de auxiliar produtores na tomada de decisões mais assertivas [Bouri et al. 2023].

O Manejo Integrado de Pragas (MIP) consiste em um conjunto de práticas que visam otimizar a produção agrícola por meio do controle eficiente e sustentável das pragas

[Abbas et al. 2022]. O MIP tem como base o monitoramento contínuo das populações de insetos-praga e o uso de critérios técnicos que orientam a tomada de decisão quanto à necessidade e ao momento ideal de intervenção.

De forma geral, o monitoramento dos insetos-praga é realizado por meio de inspeções em campo, com o uso de armadilhas equipadas com feromônios, que possibilitam a quantificação das pragas presentes na área cultivada [Preti et al. 2021]. No entanto, esse processo demanda tempo e mão de obra especializada, o que pode comprometer a eficiência e a escalabilidade das práticas de manejo em grandes áreas agrícolas.

A integração de dispositivos da Internet das Coisas (*Internet of Things*, IoT), como sensores e câmeras, com modelos de aprendizado profundo executados em sistemas embarcados tem viabilizado a detecção automática de pragas diretamente no campo, possibilitando análises mais rápidas, precisas e alinhadas às restrições operacionais dos ambientes agrícolas [Leybourne et al. 2025].

Para executar os modelos em dispositivos IoT, torna-se necessário adequar as arquiteturas de aprendizado profundo às limitações de processamento, memória e energia desses ambientes. Nesse contexto, técnicas de otimização, como quantização e poda, têm sido utilizadas para reduzir o uso de memória e o tempo de processamento, tornando viável a execução local dos modelos em campo sem perda significativa de desempenho [Ahmad et al. 2023].

Neste trabalho, são avaliadas técnicas de compressão de modelos aplicadas à arquitetura YOLO11n para a detecção da lagarta-do-cartucho (*Spodoptera frugiperda* (J.E. Smith) (*Lepidoptera: Noctuidae*)) na fase adulta, uma praga polífaga que ataca diversas culturas como o milho, soja e algodão. Os modelos foram implementados em um Raspberry Pi Zero, dispositivo IoT de baixo custo que realiza a inferência localmente, sem dependência de conectividade com servidores em nuvem, viabilizando o monitoramento em tempo real no campo. Foram analisadas métricas de desempenho e de consumo de recursos computacionais para demonstrar a viabilidade da execução em *hardware* restrito. Diferente de abordagens que modificam a arquitetura da rede ou combinam múltiplas técnicas de otimização, este trabalho investiga o potencial da quantização como estratégia isolada e de baixo custo de implementação para possibilitar sistemas de monitoramento agrícola embarcado, acessíveis a pequenos e médios produtores rurais.

2. Trabalhos Relacionados

Com a mudança para modelos de agricultura sustentável, foram elaborados métodos eficientes de controle de pragas, diminuindo o uso de produtos químicos [Velten et al. 2015]. Nesse contexto, os autores em [He et al. 2019] discutem como a policultura auxilia no controle biológico, enquanto em [Stenberg 2017] é proposto um *framework* para o Manejo Integrado de Pragas. Com foco na espécie *Spodoptera frugiperda*, uma das pragas mais devastadoras, capaz de atacar diversos cultivos, como milho, sorgo e algodão, estudos detalham sua biologia e o impacto econômico severo em diversas culturas [Kenis et al. 2022, Overton et al. 2021]. Dessa forma, o auxílio de dispositivos IoT para o monitoramento de pragas em campo é visto como uma solução promissora para o cultivo inteligente, auxiliando nas intervenções de forma eficiente[Karunathilake et al. 2023]. A Tabela 1 apresenta uma comparação dos trabalhos relacionados, quanto ao modelo utilizado, às técnicas de otimização aplicadas e a aplicação.

Tabela 1. Comparação dos trabalhos relacionados.

Trabalho	Modelo	Técnica principal	Contexto / aplicação
[Altaie et al. 2025]	YOLO11	Quantização (FP32 → INT8)	Revisão sobre otimização para dispositivos IoT
[Sharma et al. 2025]	YOLO11	Seleção de modelo reduzido	Implementação em Raspberry Pi Zero para monitoramento em tempo real
[Lin et al. 2025]	YOLO11-ARAF	Destilação de conhecimento	Detecção da *Spodoptera frugiperda* em pomares
[Wang et al. 2023]	Q-YOLO	Quantização INT8 + otimização de fluxo de dados	Detecção de objetos em larga escala para sistemas IoT
[Rai et al. 2024]	YOLO-Spot	Poda, reparametrização de módulos e quantização FP16	Identificação de ervas daninhas em imagens aéreas de drone
[Aaron et al. 2023]	InceptionV3	Quantização INT8	Identificação de ervas daninhas em plantações de milho e soja
Proposto	**YOLO11n**	**Quantização FP16 e INT8**	**Detecção da *Spodoptera frugiperda* em plantações de milho**

Nesse contexto, [Altaie et al. 2025] apresenta uma revisão sobre a otimização da família YOLO11 para dispositivos IoT. A pesquisa aponta que a compressão dos modelos via quantização é essencial para reduzir a bitagem de FP32 para INT8, permitindo que modelos funcionem com apenas 25% do consumo de memória original. Além disso, em [Sharma et al. 2025], os autores validam a implementação do YOLO11 em Raspberry Pi Zero, mostrando que a escolha de modelos reduzidos é o fator importante para manter a latência em níveis aceitáveis para o monitoramento em tempo real.

A pesquisa de [Lin et al. 2025] introduz o modelo YOLO11-ARAF, usando a arquitetura YOLO11 para detecção de *Spodoptera frugiperda* em pomares. A pesquisa usou uma técnica chamada destilação de conhecimento. Essa técnica transfere a inteligência de um modelo grande e forte para um modelo menor. Essa abordagem gerou um crescimento de 100% na velocidade de inferência (FPS), confirmando a aplicabilidade do YOLO11 para usos em dispositivos restritos.

Por outro lado, ao explorar o Q-YOLO em [Wang et al. 2023], o modelo demonstra que a quantização para INT8 pode reduzir o número de parâmetros em 75% com uma redução na precisão de apenas 0,1%. Ao contrário de outras técnicas, esta técnica otimiza o fluxo de dados, acelerando a detecção em até 31%. Dessa forma, o uso de precisão como FP16 ou INT8 mostra-se promissor para viabilizar sistemas IoT em larga escala.

Nessa mesma linha de otimização, a pesquisa de [Rai et al. 2024] introduz o modelo YOLO-Spot, baseado na arquitetura YOLOv7-tiny, voltado para a identificação de ervas daninhas em imagens aéreas capturadas por drones. O estudo utilizou as técnicas de poda (pruning), reparametrização de módulos e quantização no formato FP16. Essa abordagem gerou uma redução de mais de 75% no número de parâmetros e proporcionou uma velocidade de inferência 5 vezes maior quando implantada em um dispositivo de borda.

Por fim, [Aaron et al. 2023] apresentam um modelo focado na identificação de ervas daninhas em plantações de milho e soja utilizando a arquitetura InceptionV3, reduzindo o número de parâmetros com quantização no formato INT8. Com essa aplicação, foi possível comprimir o tamanho do modelo original de 183,34 MB para apenas 23,38 MB, mantendo uma acurácia de 87%, sendo adequado para ser instalado em dispositivos restritos.

Comparando com os trabalhos relacionados, este trabalho se destaca ao conduzir uma comparação entre a quantificação FP16 e INT8, considerando diversas métricas. Enquanto a literatura foca na redução da dimensão do modelo, esta investigação ressalta a interação entre a manutenção das métricas de detecção e o custo computacional constante Ao analisar o impacto da quantização no desempenho em dispositivos IoT, o estudo se destaca por determinar o momento em que a simplificação da complexidade começa a afetar a confiabilidade das previsões em *hardware* econômico, preenchendo uma lacuna quanto à eficiência.

3. Metodologia

A Figura 1 apresenta a metodologia proposta dividida em quatro etapas: Base de dados, Detecção de pragas, Otimização e Avaliação. Inicialmente, na etapa de Base de dados, realizou-se a aquisição de imagens contendo a *Spodoptera frugiperda*, seguida do pré-processamento, que incluiu a anotação das imagens, o redimensionamento e a divisão dos dados. Em seguida, na etapa de Detecção de pragas, foi conduzido o treinamento do modelo YOLOv11n, selecionado por sua leveza e eficiência, com ajustes de hiperparâmetros e monitoramento das métricas de desempenho ao longo das épocas. Na etapa seguinte, foram aplicadas técnicas de otimização visando reduzir o custo computacional, utilizando quantização em FP16 e INT8 para diminuir o tamanho do modelo e acelerar a inferência. Por fim, na etapa de Avaliação, os modelos foram implementados em um dispositivo IoT Raspberry Pi Zero, no qual foram realizadas análises de desempenho considerando métricas como tempo de inferência, consumo de memória e precisão, a fim de verificar a viabilidade da solução em cenários reais de monitoramento agrícola embarcado.

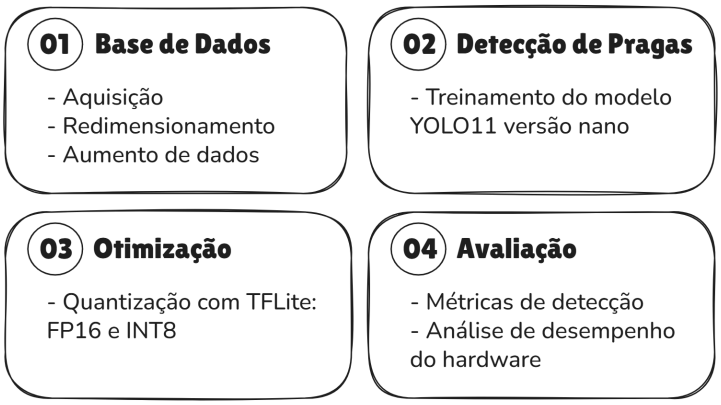

Figura 1. Descrição da metodologia proposta.

3.1. Base de Dados

A base de dados utilizada neste estudo é derivada de um conjunto desenvolvido para o monitoramento da praga *Spodoptera frugiperda* obtidas por [Silva et al. 2024]. As imagens foram coletadas por meio de uma armadilha inteligente equipada com câmera e integrada a um dispositivo embarcado, no formato RGB, com resolução de 1280×720 pixels, operando em condições reais de campo em uma lavoura de milho infestada. Para aumentar a diversidade visual e estrutural do conjunto, foram capturadas imagens adicionais em ambiente laboratorial.

Para o treinamento dos modelos, um especialista selecionou 944 amostras e realizou as anotações indicando a posição dos insetos manualmente, através de marcações das caixas delimitadoras em um arquivo .txt por imagem, utilizando a ferramenta CVAT.ai[1]. As imagens apresentam variações significativas de iluminação, ruído e densidade de insetos, contendo entre 1 e 12 insetos.

A resolução foi ajustada para 512×512 pixels, equilibrando a preservação de detalhes visuais relevantes com a redução do custo computacional. A Figura 2 apresenta um exemplo da imagem após o redimensionamento, com a caixa delimitadora destacando o inseto. Posteriormente, técnicas de aumento de dados foram aplicadas, resultando em um conjunto expandido de 1274 imagens, mantendo os conjuntos de validação e de teste apenas com imagens originais. Assim, o conjunto de treino é composto por 1020 amostras (80%), enquanto os conjuntos de validação e teste são formados por 127 amostras (10%) respectivamente. Essa divisão foi realizada para garantir que o modelo fosse treinado com uma quantidade adequada de dados, ao mesmo tempo em que se mantivesse um conjunto independente para validação e avaliação final do desempenho.

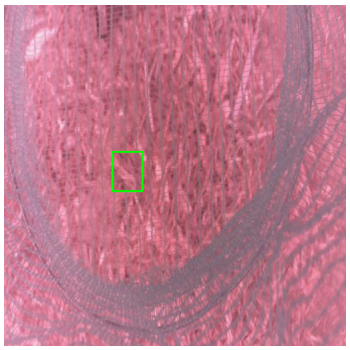

Figura 2. Exemplo de imagem da base de dados: o retângulo verde representa a anotação usada para detectar os insetos.

3.2. Detecção de Pragas

Para a tarefa de detecção de pragas, foi utilizado o modelo YOLO11[2] na versão nano (n), com 2.6M de parâmetros, uma arquitetura de detecção de objetos em estágio único (*single-stage detector*), que realiza simultaneamente a regressão das caixas delimitadoras e a classificação dos objetos, permitindo detecção eficiente com baixa latência [Sayyad et al. 2025].

O modelo foi treinado para identificar exclusivamente a classe correspondente a *Spodoptera frugiperda*, utilizando as anotações da base de dados descrita anteriormente, e após a análise de gráficos de desempenho no treinamento com o conjunto de validação, foi definido que 50 épocas e o otimizador SGD (*Stochastic Gradient Descent*) foram os parâmetros que apresentaram melhor equilíbrio entre desempenho e estabilidade do treinamento. Observou-se que, a partir de 50 épocas, a redução da função de perda tornava-se pouco significativa, indicando convergência do modelo e evitando o risco de sobreajuste. Além disso, o otimizador SGD demonstrou maior estabilidade na atualização dos pesos

[1]https://github.com/cvat-ai/cvat
[2]https://platform.ultralytics.com/ultralytics/yolo11

quando comparado a outros otimizadores testados, proporcionando uma curva de aprendizado mais consistente.

A escolha do YOLO11n se justifica por seu bom desempenho em aplicações embarcadas e pela compatibilidade com técnicas de otimização e quantização oferecidas pelo TensorFlow Lite, as quais permitem a redução do tamanho do modelo e do consumo de memória, além de diminuir o tempo de inferência [Sayyad et al. 2025, Wang et al. 2025].

3.3. Otimização

O modelo foi embarcado em um dispositivo Raspberry Pi Zero 2W para execução e validação em ambiente embarcado. Visando reduzir o consumo de recursos computacionais e viabilizar a execução do modelo em dispositivos com restrições de hardware, foram aplicadas técnicas de quantização utilizando o TensorFlow Lite. As representações numéricas avaliadas são descritas na Tabela 2.

Tabela 2. Representações numéricas utilizadas no processo de quantização

Representação	Precisão	Descrição
FP32	32 bits	Representação em ponto flutuante utilizada como linha de base, apresentando maior precisão numérica e maior custo computacional.
FP16	16 bits	Reduz o consumo de memória e acelera as operações aritméticas, mantendo boa precisão.
INT8	8 bits	Quantização inteira que proporciona maior compressão do modelo e menor latência, com possível degradação na acurácia.

A quantização pode ser formalmente descrita pela Equação 1:

$$q = \text{round}\left(\frac{x}{s}\right) + z \tag{1}$$

onde x representa o valor original em ponto flutuante, q é o valor quantizado, s é o fator de escala (*scale*) e z é o ponto zero (*zero-point*). A reconstrução aproximada do valor original é dada pela Equação 2:

$$\hat{x} = s \cdot (q - z) \tag{2}$$

No caso da quantização INT8, essa transformação é aplicada tanto aos pesos quanto às ativações da rede, permitindo execução eficiente em hardware que suporta operações inteiras. Para FP16, a redução ocorre diretamente na precisão dos pesos e das ativações, sem a necessidade de fatores de escala explícitos.

3.4. Avaliação

A avaliação do modelo considerou métricas de desempenho relacionadas à detecção de objetos, bem como indicadores de eficiência computacional, refletindo o compromisso entre acurácia e custo de execução. As métricas de detecção são apresentadas a seguir:

- mAP@0.5: Média da precisão média considerando um limiar de interseção sobre união (IoU) igual a 0,5.
- mAP@0.5:0.95: Média do mAP para limiares de IoU variando de 0,5 a 0,95, com incremento de 0,05.
- Precisão: Proporção de detecções corretas em relação ao total de detecções realizadas.
- Recall: Proporção de objetos corretamente detectados em relação ao total de objetos presentes.

Além das métricas de detecção, foram coletadas métricas de desempenho relacionadas ao hardware, como tempo de inferência, uso de memória RAM, temperatura, utilização da CPU e consumo de energia, incluindo medições de corrente, potência e tensão. Para a captura das variáveis elétricas de tensão, corrente e potência contínua (DC), utilizou-se o módulo de monitoramento bidirecional de alta precisão INA219. Embora o referido sensor tenha compartilhado a mesma fonte de baterias utilizada pelo Raspberry Pi Zero 2W, sua alimentação foi implementada de forma independente. Dessa forma, o sensor não foi alimentado diretamente pelo Raspberry Pi, garantindo que o seu próprio consumo de operação não fosse contabilizado nos dados energéticos do sistema. Ademais, os dados de temperatura e uso de recursos computacionais (CPU e RAM) foram obtidos por meio dos sensores internos integrados ao dispositivo de borda. Essas informações permitem verificar se o modelo atende aos requisitos de execução em tempo real, considerando as limitações de recursos computacionais.

4. Experimentos e Resultados

Nesta seção, são analisados os resultados dos experimentos que avaliaram o impacto da otimização de modelos de detecção em um dispositivo IoT com recursos computacionais restritos. Para isso, foram conduzidos testes comparando três níveis de precisão numérica: ponto flutuante de 32 bits (FP32, adotado como referência), ponto flutuante de 16 bits (FP16) e quantização inteira de 8 bits (INT8), conforme sintetizado na Tabela 3.

Observou-se que as três variantes INT8, FLOAT16 e FLOAT32 apresentaram comportamento equivalente quanto ao desempenho de detecção, com as 127 imagens do conjunto de teste. As métricas de precisão mantiveram-se elevadas em todas as configurações, com valores superiores à 0,98 para a Precisão, Recall e mAP@0.5 . O mAP@0.5:0.95 permaneceu similar entre FP32 e FP16 com 0,81, com uma redução de 0,80 para o modelo INT8.

Embora a quantização INT8 tenha apresentado uma pequena redução nas métricas de detecção em relação aos modelos FP16 e FP32, o impacto observado foi relativamente baixo, indicando que a redução da precisão numérica não comprometeu de forma significativa a capacidade do modelo de identificar os insetos. Diferentes estratégias podem ser empregadas para mitigar esse impacto na taxa de acerto, como o uso de Quantization-Aware Training (QAT), no qual o modelo é treinado considerando previamente os efeitos

Tabela 3. Resumo dos resultados do YOLO11n no hardware-alvo para diferentes precisões numéricas

Métrica	INT8	FP16	FP32
Precisão	0,991	0,993	0,994
Recall	0,982	0,987	0,991
mAP@0.5	0,991	0,993	0,994
mAP@0.5:0.95	0,805	0,811	0,812
Inferência média por imagem (s)	1,451	1,217	1,239
Inferência, desvio padrão (s)	0,434	0,504	0,671
Inferência máxima (s)	6,310	6,860	8,744
Temperatura média da CPU (° C)	54,387	56,136	57,318
Uso médio de CPU (%)	0,416	0,284	0,285
Uso máximo de CPU (%)	2,5	2,5	4,8
Uso médio de RAM (%)	73,406	73,746	66,611
Corrente média (mA)	295,100	326,117	334,900
Tensão média (V)	5,140	5,139	5,138
Potência média	1516,765	1676,023	1720,745

da quantização, além da aplicação de técnicas de poda estruturada para reduzir a complexidade computacional preservando a qualidade das predições.

No que se refere à inferência dos modelos, verificou-se uma diferença mais nítida entre as variantes. O modelo FP16 apresentou o menor tempo médio por imagem, com 1,21 s seguido do FP32, com 1,23 s e desvio padrão de 0,6713 s, e do INT8, com 1,45 s e desvio padrão de 0,4346 s. Apesar do INT8 ser frequentemente associado a ganhos de desempenho, os resultados indicaram maior latência média e um intervalo máximo elevado, com 6,31 s, em comparação ao FP16, com 6,86 s, e ao FLOAT32, com 8,74 s. Os resultados indicam dispersão no tempo de inferência, com ocorrência de picos pontuais de latência. Esse comportamento é decorrente da arquitetura de hardware utilizada, que não possui instruções otimizadas para cálculos de inteiros de 8 bits, tornando necessários mais ciclos de processamento para realizar as operações do modelo.

Quanto ao comportamento térmico e computacional, observou-se que a temperatura média da CPU aumentou com a precisão, já o uso médio de CPU permaneceu baixo em todas as variantes, com valores médios inferiores a 0,42 %, porém com máximos que indicam picos ocasionais, atingindo até 2,5% nos modelos com quantização e até 4,8% no modelo FP32. Em termos de memória, os modelos INT8 e FP16 mantiveram uso de RAM semelhante, aproximadamente em 73 % e cerca de 305 a 307 MB, enquanto o FP32 apresentou menor uso médio de RAM, com 66,6118 % e 277,2866 MB, ainda que com maior amplitude de variação.

Sob a perspectiva do consumo de energia, observou-se que a variante INT8 esteve associada aos menores valores médios de corrente e potência. A corrente média foi de 295,1003 mA no INT8, aumentando para 326,1170 mA no FP16 e 334,9009 mA no FP32. As potências médias acompanharam essa tendência, com potência igual a 1516,7652 mW no INT8, 1676,0233 mW no FP16 e 1720,7458 mW no FP32.Por fim, a tensão média também manteve-se praticamente constante entre as três configurações, com

valores próximos de 5,14 V, indicando estabilidade da alimentação elétrica do sistema.

De forma geral, os resultados demonstram que a escolha da precisão numérica impacta principalmente no consumo de recursos computacionais e de energia, enquanto o desempenho de detecção permanece praticamente equivalente entre as três configurações avaliadas. O modelo INT8 apresentou menor consumo de corrente e potência, além de menor temperatura média da CPU, indicando maior eficiência energética, característica relevante para aplicações embarcadas e sistemas com restrições de energia.

5. Considerações Finais

Neste trabalho, foi possível demonstrar que o uso de dispositivos IoT aplicados à agricultura não depende apenas da escolha das arquiteturas, mas de como equilibrar a precisão dos modelos com as limitações do *hardware*. Os resultados obtidos evidenciam que modelos embarcados podem operar de forma eficiente no contexto agrícola, tornando viável sua aplicação em sistemas de monitoramento contínuo para o Manejo Integrado de Pragas.

Os experimentos com o modelo YOLO11n demonstraram que a quantização INT8 manteve desempenho competitivo em relação às representações FP16 e FP32, preservando a qualidade das predições mesmo com menor custo computacional. Embora tenha apresentado um tempo médio de inferência maior, o formato INT8 destacou-se pela eficiência na energia e na estabilidade operacional, apresentando menor consumo de potência, corrente elétrica e temperatura da CPU em comparação às demais precisões numéricas. Esses resultados evidenciam o potencial do INT8 para aplicações na agricultura, favorecendo maior autonomia energética dos dispositivos embarcados e viabilizando seu uso em ambientes rurais com infraestrutura limitada.

Além disso, o menor consumo de energia, aliado à manutenção da qualidade das predições, favorece o monitoramento de pragas no campo, permitindo a detecção precoce e o suporte à tomada de decisão em tempo real. Isso pode auxiliar produtores rurais na redução de perdas agrícolas e de custos operacionais, fortalecendo o uso de estratégias sustentáveis.

Apesar dos resultados promissores, este trabalho apresenta algumas limitações. Os experimentos foram conduzidos em um único hardware-alvo, o que limita a generalização dos resultados para outras plataformas embarcadas. Além disso, foram analisadas apenas técnicas relacionadas à representação numérica dos modelos, sem explorar outras estratégias de otimização.

Em virtude disso, os próximos trabalhos precisam ir além da compressão de pesos, direcionando-se à redução da complexidade dos modelos em si. Ao mesmo tempo, e tendo em vista os bons resultados da representação INT8, seria interessante examinar o funcionamento do sistema utilizando exclusivamente números inteiros, medindo a possível redução adicional no consumo de energia ao desligar por completo a Unidade de Ponto Flutuante (FPU) do processador. É interessante também investigar o desempenho dos modelos em microcontroladores que possuam Unidades de Processamento Neural (NPUs) para permitir o monitoramento em tempo real com altas taxas de quadros, estabelecendo esses sistemas como soluções essenciais no controle integrado de pragas. Além disso, podem ser investigadas outras técnicas de otimização, como a poda de rede e a destilação de conhecimento, que podem resultar em melhorias adicionais de desempenho e eficiência energética.

Referências

Aaron, A., Hassan, M., Hamada, M., and Kakudi, H. (2023). A lightweight deep learning model for identifying weeds in corn and soybean using quantization. *Engineering Proceedings*, 56(1):318.

Abbas, M., Saleem, M., Hussain, D., Ramzan, M., Jawad Saleem, M., Abbas, S., Hussain, N., Irshad, M., Hussain, K., Ghouse, G., et al. (2022). Review on integrated disease and pest management of field crops. *International Journal of Tropical Insect Science*, 42(5):3235–3243.

Ahmad, S., Shakeel, I., Mehfuz, S., and Ahmad, J. (2023). Deep learning models for cloud, edge, fog, and iot computing paradigms: Survey, recent advances, and future directions. *Computer Science Review*, 49:100568.

Altaie, U. K., Abdelkareem, A. E., and Alhasanat, A. (2025). Lightweight optimization of yolo models for resource-constrained devices: A comprehensive review. *Diyala Journal of Engineering Sciences*, 18(4):1–26.

Bouri, M., Arslan, K. S., and Şahin, F. (2023). Climate-smart pest management in sustainable agriculture: Promises and challenges. *Sustainability*, 15(5):4592.

Ediagbonya, T. F., Areo, I. O., Mupenzi, C., Mind'je, R., Kamuhanda, J. K., and Kabano, S. (2025). Reduced pesticide dependency through crop management. *Discover Applied Sciences*, 7(7):776.

He, H.-m., Liu, L.-n., Munir, S., Bashir, N. H., Yi, W., Jing, Y., and Li, C.-y. (2019). Crop diversity and pest management in sustainable agriculture. *Journal of Integrative Agriculture*, 18:1945–1952.

Karunathilake, E., Le, A. T., Heo, S., Chung, Y. S., and Mansoor, S. (2023). The path to smart farming: Innovations and opportunities in precision agriculture. *Agriculture*, 13:1593.

Kenis, M., Benelli, G., Biondi, A., Calatayud, P.-A., Day, R., Desneux, N., Harrison, R. D., Kriticos, D., Rwomushana, I., van den Berg, J., et al. (2022). *Invasiveness, biology, ecology, and management of the fall armyworm, Spodoptera frugiperda*. Schweizerbart Science Publishers.

Leybourne, D. J., Musa, N., and Yang, P. (2025). Can artificial intelligence be integrated into pest monitoring schemes to help achieve sustainable agriculture? an entomological, management and computational perspective. *Agricultural and Forest Entomology*, 27(1):8–17.

Lin, Y., Xia, Y., Xia, P., Liu, Z., Wang, H., Qin, C., Gong, L., and Liu, C. (2025). Yolo11-araf: An accurate and lightweight method for apple detection in real-world complex orchard environments. *Agriculture*, 15(10):1104.

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

Preti, M., Verheggen, F., and Angeli, S. (2021). Insect pest monitoring with camera-equipped traps: strengths and limitations. *Journal of pest science*, 94(2):203–217.

Rai, N., Zhang, Y., Villamil, M., Howatt, K., Ostlie, M., and Sun, X. (2024). Agricultural weed identification in images and videos by integrating optimized deep learning architecture on an edge computing technology. *Computers and Electronics in Agriculture*, 216:108442.

Sayyad, J., Attarde, K., and Almustafa, K. M. (2025). A systematic literature review on deep learning approaches for small object detection. *Array*, page 100615.

Sharma, R., Sharma, B. B., et al. (2025). Design and implementation of real-time tomato plant growth monitoring system using deep learning based yolo and raspberry pi. *Applied Artificial Intelligence*.

Silva, W., Soares, B., Almeida, V., Viana, L., Pastori, P., Magalhães, D., and Rocha, A. (2024). Detecção da praga spodoptera frugiperda no cultivo de milho usando armadilhas inteligentes e visão computacional. In *Anais do XV Workshop de Computação Aplicada à Gestão do Meio Ambiente e Recursos Naturais*, pages 61–70, Porto Alegre, RS, Brasil. SBC.

Stenberg, J. A. (2017). A conceptual framework for integrated pest management. *Trends in Plant Science*, 22:759–769.

Velten, S., Leventon, J., Jager, N., and Newig, J. (2015). What is sustainable agriculture? A systematic review. *Sustainability*, 7:7833–7865.

Wang, C., Han, Y., Yang, C., Wu, M., Chen, Z., Yun, L., and Jin, X. (2025). Cf-yolo for small target detection in drone imagery based on yolov11 algorithm. *Scientific reports*, 15(1):16741.

Wang, M. et al. (2023). Q-yolo: Efficient inference for real-time object detection. *Pattern Recognition (ACPR 2023)*, 14408.