Previsão do nível de um tanque em tempo real utilizando IA embarcada em um ESP32-CAM-S3 através da plataforma Edge Impulse e TensorFlow lite

1st Gustavo Guilherme Wanderley
Departamento de Engenharia da Computação (DCA)
Universidade Federal do Rio Grande do Norte (UFRN)
Natal, Brasil
gustavo.wanderley.703@ufrn.edu.br

2nd Wellington Silva de Oliveira
Departamento de Engenharia da Computação (DCA)
Universidade Federal do Rio Grande do Norte (UFRN)
Natal, Brasil
wellington.oliveira.708@ufrn.edu.br

3rd Diego Medeiros Ponte
Departamento de Engenharia da Computação (DCA)
Universidade Federal do Rio Grande do Norte (UFRN)
Natal, Brasil
diego.ponte.114@ufrn.edu.br

4th Heitor Medeiros Florêncio
Instituto Metrópole Digital (IMD)
Universidade Federal do Rio Grande do Norte (UFRN)
Natal, Brasil
heitorm@imd.ufrn.br

5th Daniel Lopes Martins
Instituto Metrópole Digital (IMD)
Universidade Federal do Rio Grande do Norte (UFRN)
Natal, Brasil
daniel.martins@imd.ufrn.br

6th Adrião Duarte Dória Neto
Departamento de Engenharia da Computação (DCA)
Universidade Federal do Rio Grande do Norte (UFRN)
Natal, Brasil
adriao@dca.ufrn.br

***Abstract*—This paper presents the implementation of an embedded computer vision model on the ESP32-CAM microcontroller to measure a tank's level. The system uses the Edge Impulse platform for model training and TensorFlow Lite for its execution. The model, based on convolutional neural networks, was quantized to ensure efficiency on resource-constrained devices. The AI system's performance is rigorously evaluated through a direct comparison with a high-precision industrial level sensor, validating its accuracy and identifying its limitations in a practical scenario.**

***Index Terms*—IA, GPU, Quantização, ESP32, Visão Computacional, Edge Impulse, TensorFlow Lite.**

I. Introdução

A medição de nível de tanques é um pilar para a automação industrial, onde sensores de alta precisão, como os de pressão e radar, são essenciais para garantir segurança e eficiência. Essas tecnologias são a base para o controle de processos críticos, oferecendo alta confiabilidade na maioria das aplicações.

Contudo, existem cenários operacionais específicos onde até mesmo esses sensores robustos enfrentam desafios. Um sensor de pressão, por ser inerentemente invasivo, pode sofrer com entupimentos ou se degradar rapidamente com produtos químicos, tornando-se um ponto de falha [2]. Já os sensores de radar, apesar de sua excelência, podem ser temporariamente "cegados" por uma camada de espuma muito densa ou por um material que absorva as ondas eletromagnéticas, limitações conhecidas na literatura [9].

É para preencher essas lacunas que a medição por câmera e Inteligência Artificial se apresenta como uma solução estratégica e complementar. O sistema utiliza uma câmera de baixo custo para "ler" um indicador de nível externo, funcionando como um par de olhos digital, independente e verificador. Por não ter contato com o produto, é imune a entupimentos e corrosão.

O verdadeiro impacto desta tecnologia está em sua capacidade de trabalhar em conjunto com os sistemas existentes. Ela pode atuar como um sistema de redundância de baixo custo, validando as leituras do sensor principal e alertando sobre divergências que possam indicar uma falha iminente. Adicionalmente, permite expandir a automação para tanques auxiliares de forma econômica, garantindo uma visão completa e mais segura de toda a planta industrial [2].

Com o avanço da Indústria 4.0 e da Internet das Coisas (IoT), surge uma crescente demanda por soluções de monitoramento que sejam mais flexíveis, de baixo custo e não invasivas. Neste contexto, a aplicação de Inteligência Artificial (IA) embarcada em microcontroladores de baixo custo apresenta-se como uma alternativa promissora. A capacidade de inferir uma variável física, como o nível, a partir de imagens capturadas em tempo real, elimina a necessidade de contato direto com o

fluido, sendo viabilizada por frameworks como o TensorFlow Lite, projetado especificamente para essa classe de dispositivos [1].

A relevância deste tipo de pesquisa é acentuada pela atual necessidade de três tendências tecnológicas. Primeiro, a presença de microcontroladores potentes e acessíveis, como a série ESP32, que integram processamento, conectividade e interfaces de câmera a um custo acessível. Segundo, a maturidade de frameworks de software, como o TensorFlow Lite, que são bem otimizados para hardware com recursos restritos. Terceiro, o surgimento de plataformas, como a Edge Impulse, que facilitam o uso e aceleram o ciclo de desenvolvimento da IA embarcada. É a união destes fatores que torna a presente proposta não apenas viável, mas também torna possível solucionar muitos problemas com esta tecnologia.

Apesar desses pontos positivos, a validação rigorosa de tais sistemas de baixo custo contra padrões industriais consolidados é um passo crucial para garantir sua confiabilidade no chão de fábrica. Portanto, o objetivo central deste trabalho não é apenas demonstrar a viabilidade técnica da solução, mas, fundamentalmente, avaliar seu desempenho de forma quantitativa, comparando diretamente as medições da IA com as de um sensor de nível industrial de alta precisão.

II. Fundamentação Teórica

Para contextualizar a solução desenvolvida, esta seção aborda as metodologias de medição, os conceitos de software e o hardware que viabilizam a implementação da inteligência artificial.

A. Metodologias de Medição de Nível

1) Abordagem Industrial com Sensores Dedicados: A medição de nível em ambientes industriais é classicamente realizada por meio de instrumentação dedicada, projetada para alta precisão e robustez. Exemplos comuns incluem sensores ultrassônicos que funcionam emitindo um pulso sonoro e medindo o tempo que esse pulso leva para refletir de volta após atingir a superfície desejada. A distância é calculada com base nesse tempo de voo, considerando a velocidade do som no meio em questão. Outro exemplo são os sensores de radar que operam de maneira semelhante, mas utilizam ondas eletromagnéticas em vez de som. Por conta disso, eles são menos suscetíveis a interferências ambientais e oferecem maior precisão e alcance, além de funcionarem melhor em ambientes com vapor, poeira ou pressões extremas [9]. Outra abordagem comum, e a utilizada como referência neste trabalho, emprega sensores de pressão estática (manométrica) instalados na base do tanque. Estes sensores medem a pressão exercida pela coluna de líquido para então inferir a altura [2]. O sistema de referência ilustrado na Figura 1, representa essa abordagem convencional, utilizando sensores industriais (Sensor Nível 1 e 2) para obter leituras diretas e precisas.

2) Abordagem Proposta com Visão Computacional: Como alternativa, este estudo propõe uma abordagem não invasiva e de baixo custo que utiliza visão computacional para inferir o nível do tanque. A estrutura experimental para esta metodologia é esquematizada na Figura 2. Um microcontrolador com câmera, o ESP32-CAM-S3, é posicionado externamente ao tanque, utilizando seu campo de visão para capturar imagens de um indicador visual de nível. Essas imagens são pré-processadas localmente pelo próprio hardware do ESP32-CAM-S3, de forma a adequá-las aos parâmetros exigidos pelo modelo. Em seguida, um modelo de inteligência artificial embarcado realiza a inferência do nível do tanque com base nas imagens processadas, eliminando a necessidade de sensores físicos dedicados à medição de nível.

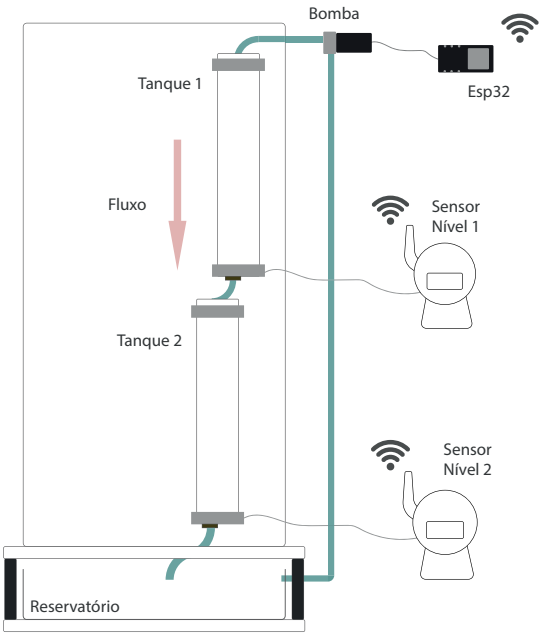

Fig. 1. Planta do sistema de recalque, ilustrando o uso de sensores de nível industriais dedicados.

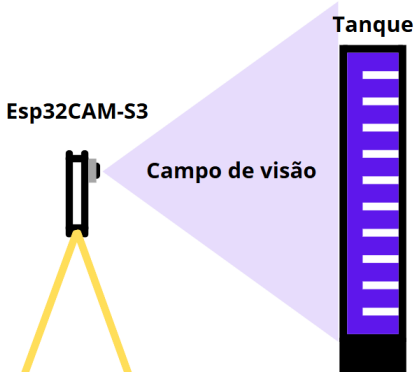

Fig. 2. Esquemático da estrutura proposta, com o ESP32-CAM-S3 medindo o nível do tanque através de seu campo de visão.

B. Visão Computacional e Redes Neurais Convolucionais

A Visão Computacional é um campo da IA que busca capacitar os computadores a enxergar e interpretar o mundo visual. As tarefas incluem classificação de imagens, detecção

de objetos e, como neste trabalho, regressão para estimar um valor a partir de uma imagem [3]. As Redes Neurais Convolucionais (CNNs) são a arquitetura de deep learning predominante para essas tarefas.

Uma CNN é tipicamente composta por três tipos de camadas: convolucionais, de pooling e totalmente conectadas.

A camada convolucional é o núcleo de uma CNN. Nela, utilizamos filtros (kernels) que são pequenas matrizes com valores numéricos. Esses filtros percorrem a imagem de entrada, posição por posição, realizando uma operação chamada produto escalar (ou produto de ponto) entre os valores do filtro e os valores da imagem naquela região. Esse processo gera um mapa de características (feature map) que destaca padrões específicos na imagem, como arestas, texturas ou formas. As primeiras camadas aprendem características simples, como bordas e texturas. À medida que a rede se aprofunda, as camadas seguintes, geralmente compostas por um número maior de filtros, combinam essas informações para identificar padrões mais complexos. Isso ocorre porque uma CNN pode conter várias camadas convolucionais empilhadas, e cada camada extrai representações cada vez mais abstratas a partir das ativações da anterior. Dessa forma, a rede é capaz de construir gradualmente uma compreensão mais rica da imagem, indo de elementos básicos até estruturas mais complexas. [4].

Inserida entre as camadas convolucionais, a camada de pooling (geralmente Max Pooling) tem o objetivo de reduzir a dimensão dos mapas de características. Ela opera sobre pequenas janelas do mapa e substitui os valores daquela janela por um único valor (o máximo, no caso do Max Pooling). Isso torna a representação mais robusta a pequenas translações na imagem e reduz a carga computacional da rede [4].

Após as sucessivas etapas de convolução e pooling, os dados são processados por uma camada de achatamento, conhecida como Flatten. A função desta camada é transformar a estrutura de dados multidimensional (por exemplo, uma matriz de características de 2D) em um vetor unidimensional. Essa reorganização é um passo fundamental para que os dados possam ser utilizados pela rede, composta por uma ou mais camadas totalmente conectadas (*Fully Connected* ou *Dense*), onde cada neurônio se conecta a todos os neurônios da camada anterior [4].

Estas camadas densas finais funcionam como a parte decisória da rede, atuando como um classificador ou regressor que utiliza as características extraídas para realizar a predição. A configuração da última camada densa, especialmente sua função de ativação, define a natureza da saída do modelo:

- **Regressão (Saída Linear):** Para prever um valor contínuo, a última camada possui uma saída linear. Isso permite que a rede produza qualquer valor numérico. É o caso deste trabalho, onde o objetivo é a regressão do nível do tanque, prevendo um valor específico dentro de um intervalo contínuo.
- **Classificação Binária (Função Sigmoid):** Para problemas de classificação binária (onde a resposta é "sim" ou "não"), utiliza-se a função de ativação **Sigmoid**. Ela comprime a saída para um valor entre 0 e 1, que pode ser interpretado como a probabilidade de pertencer a uma classe.
- **Classificação Multiclasse (Função Softmax):** Para classificar uma entrada em três ou mais categorias, a função Softmax é empregada. Ela converte as saídas da camada em um vetor de probabilidades, onde a soma de todos os elementos é 1. Cada valor no vetor representa a probabilidade de a entrada pertencer a uma das classes. Por exemplo, um modelo que classifica imagens de veículos entre "carro", "moto" e "caminhão".

A arquitetura específica do modelo utilizado neste trabalho, ilustrada na Figura 3, foi desenvolvida com o auxílio da plataforma Edge Impulse. Uma característica notável e não convencional desta arquitetura é a utilização de um *kernel* de dimensões 128x128 na primeira camada convolucional. É crucial destacar que esta escolha diverge significativamente das práticas habituais em Redes Neurais Convolucionais (CNNs), que faz o uso de *kernels* pequenos (como 3x3 ou 5x5) e empilhados em várias camadas. A justificativa para a adoção deste *kernel* de grande dimensão reside no fato de que esta arquitetura é a forma padrão da ferramenta, a qual correspondeu bem em testes estáticos, somado à falta de tempo para testar outros tamanhos de kernels até a data de submissão do trabalho. Essa escolha, embora atípica, tem uma consequência direta: um *kernel* de 128x128 resulta em um número exponencialmente maior de parâmetros a serem treinados já na primeira camada, o que contribui significativamente para o tamanho final do modelo. Apesar da divergência, é importante ressaltar que o modelo gerado, mesmo com essa particularidade, conseguiu realizar a predição do nível do tanque.

C. O Framework TensorFlow Lite e a Otimização de Modelos

Historicamente, a execução de modelos de deep learning era restrita a servidores com GPUs potentes. O TensorFlow Lite (TFLite) é o framework do Google projetado para quebrar essa barreira, permitindo que modelos de IA rodem em dispositivos com recursos restritos. Sua variante, TensorFlow Lite for Microcontrollers, é otimizada para ambientes com memória na ordem de kilobytes [1].

O uso de TFLite [1]para processamento local (on-device) oferece benefícios importantes como maior privacidade, baixa latência, baixo consumo de energia e custo reduzido de hardware.

Para viabilizar a execução, os modelos precisam ser otimizados. A técnica central é a quantização, que consiste em reduzir a precisão numérica dos pesos e/ou ativações da rede. Neste trabalho, foi convertido os pesos de float32 para int8 [6]. Essa otimização reduz o tamanho do modelo em aproximadamente 4x e acelera a inferência, o que poderá ser melhor visto na seção de metodologia experimental.

D. Hardware da Solução Proposta

Para a implementação da metodologia de visão computacional (Figura 2), o hardware selecionado é um componente central. O módulo utilizado foi o **ESP32-CAM-S3**, uma placa

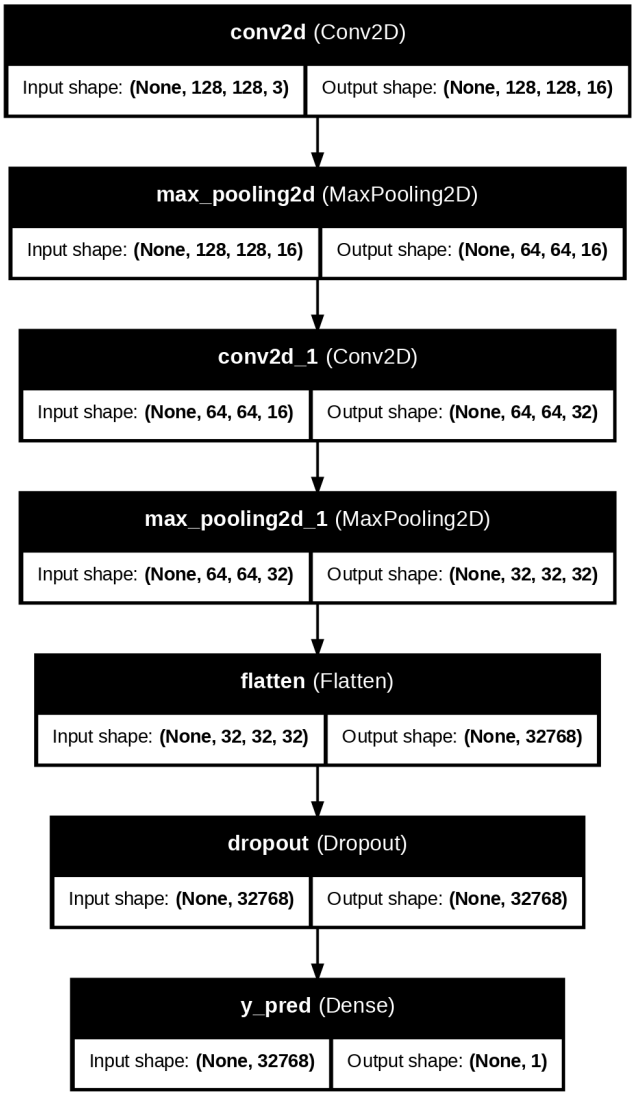

Fig. 3. Arquitetura do modelo de CNN utilizado.

de desenvolvimento que utiliza o poderoso microcontrolador ESP32-S3 [7] da Espressif Systems.

O ESP32 é um microcontrolador de baixo custo e alta performance, conhecido por sua versatilidade em projetos de Internet das Coisas (IoT). Em sua essência, ele integra um processador, memória e uma vasta gama de periféricos em um único chip. Isso inclui:

- **Pinos de I/O(GPIO):** Permitem a comunicação com sensores, atuadores e outros componentes eletrônicos.
- **Conversores Analógico-Digital (ADC) e Digital-Analógico (DAC):** Essenciais para a interface com sensores que fornecem sinais analógicos.
- **Sensores de Toque Capacitivo:** Possibilitam a criação de interfaces de usuário sensíveis ao toque.
- **Interfaces de Comunicação:** Suporte nativo para Wi-Fi e Bluetooth, além de protocolos como SPI, I2C e UART.

A versão **S3** do ESP32, utilizada neste projeto é uma evolução projetada especificamente para tarefas mais exigentes, como o uso de modelos de Inteligência Artificial. Suas principais vantagens são:

1) **Processador Dual-Core Xtensa® LX7:** Mais moderno e eficiente que o LX6 das gerações anteriores.
2) **Extensões Vetoriais (Vector Instructions):** Este é o grande diferencial. O processador LX7 inclui instruções específicas para acelerar operações matemáticas em vetores e matrizes. Como as redes neurais, base do TensorFlow Lite, dependem massivamente de cálculos matriciais (multiplicações e convoluções), essas extensões aceleram drasticamente o tempo de inferência do modelo.
3) **Boa quantidade de memória:** Com 8 MB de PSRAM e 16 MB de Flash, o ESP32-CAM-S3 possui os recursos necessários para armazenar e executar modelos de IA mais complexos, que não caberiam em microcontroladores mais simples.

A combinação de um processador mais rápido com aceleração de hardware específica para IA torna o ESP32-S3 a escolha ideal para executar modelos otimizados com o framework TensorFlow Lite.

A tabela I a seguir resume as especificações do módulo empregado.

TABLE I
ESPECIFICAÇÕES DO HARDWARE ESP32-CAM S3

Especificação	**Descrição**
Microcontrolador	ESP32-S3, Dual-core Xtensa® 32-bit LX7
Memória Flash	16 MB (SPI Flash)
PSRAM	8 MB
Câmera	OV2640 / OV5640 suportadas
Conectividade	Wi-Fi 802.11 b/g/n + Bluetooth 5.0 (LE)

III. METODOLOGIA EXPERIMENTAL

A. Construção do Dataset

A base para o modelo de IA foi um conjunto de 1240 imagens no formato RGB (320x240 pixels), capturadas com a própria câmera do ESP32-CAM-S3. Para garantir diversidade, foram coletadas 40 fotos para cada centímetro de nível do tanque, com objetivo de representar variações naturais que ocorrem durante a captura, como mudanças na iluminação ambiente, presença de ruído na imagem e pequenas variações no foco. Essas variações ajudam a tornar o modelo mais robusto, evitando que ele se ajuste apenas a condições ideais e permitindo uma melhor generalização durante a inferência. O conjunto foi dividido de forma aleatória, com 75% das imagens para treinamento e 25% para teste.

B. Treinamento e Quantização

O treinamento foi conduzido na plataforma Edge Impulse. O modelo de CNN (Figura 3) foi treinado por 150 épocas utilizando o otimizador Adam com uma taxa de aprendizado de 0.005 e um tamanho de lote (batch size) de 128. Essa parametrização foi feita com base em uma série de testes. As métricas de erro como, MSE (Erro Quadrático Médio) e o

MAE (Erro absoluto Médio) foram essenciais para encontrar esta combinação. Uma porção dos dados de treino foi usada como conjunto de validação para monitorar o desempenho e evitar sobreajuste (overfitting). Após o treinamento, o modelo foi convertido e quantizado para o formato TensorFlow Lite (TFLite) com pesos em int8. Inicialmente foi utilizado o ESP32-S [2] para tirar resultados iniciais e posteriormente, com a aquisição do ESP32-CAM-S3, o trabalho passou a utilizá-lo pelo ganho de perfomance que será apresentado nos tópicos seguintes do artigo. A seguir temos a Tabela II que representa as especificações do ESP32-S e a Tabela III representa a diferença de desempenho entre o modelo quantizado(int8) e o modelo sem otimização(float32).

TABLE II
ESPECIFICAÇÕES DO HARDWARE ESP32-S

Especificação	**Descrição**
Microcontrolador	ESP32-S, Dual-core Tensilica Xtensa® LX6
Frequência de Clock	Até 240 MHz
Memória Flash	Tipicamente 4 MB (SPI Flash no módulo)
SRAM	520 KB (Integrada ao chip)
Conectividade	Wi-Fi 802.11 b/g/n + Bluetooth v4.2

TABLE III
COMPARAÇÃO ENTRE MODELO QUANTIZADO E FLOAT32(ESP32-S)

Parâmetro	**Quantizado (int8)**	**Float32**
Latência de Inferência	1.546 ms	8.584 ms
Uso de RAM	322.6 KB	1.3 MB
Uso de Flash (ROM)	63.2 KB	168.4 KB
Acurácia (teste)	99.2%	99.8%

Como apresentado na Tabela III, temos a diferença de desempenho entre os modelos numa simulação baseada no ESP32-S [2]. Como já mencionado previamente, o resultado obtido com o chip mais potente será apresentado ao longo do artigo.

IV. RESULTADOS E DISCUSSÃO

Esta seção detalha os resultados obtidos, da análise do modelo à validação prática contra o sensor industrial.

A. Análise da Extração de Características

A Figura 4 apresenta uma visualização UMAP (Uniform Manifold Approximation and Projection) [10] das características extraídas pela CNN. A utilização do UMAP justifica-se por ser uma técnica de redução de dimensionalidade não linear, especialmente eficaz em preservar a estrutura topológica de dados complexos. Ao projetar as representações internas de alta dimensão aprendidas pela rede para um espaço bidimensional, o UMAP permite uma inspeção visual precisa da geometria do conhecimento adquirido pelo modelo.

O gráfico resultante fornece evidências claras sobre o sucesso da aprendizagem. Observa-se a formação de agrupamentos distintos, indicando que a CNN aprendeu um espaço de características onde amostras de diferentes níveis de água são altamente separáveis. Além disso, a progressão lógica que vai do vermelho (nível 0) ao azul (nível 30), demonstra que o modelo não apenas classificou, mas também ordenou as representações de maneira análoga à natureza física do problema.

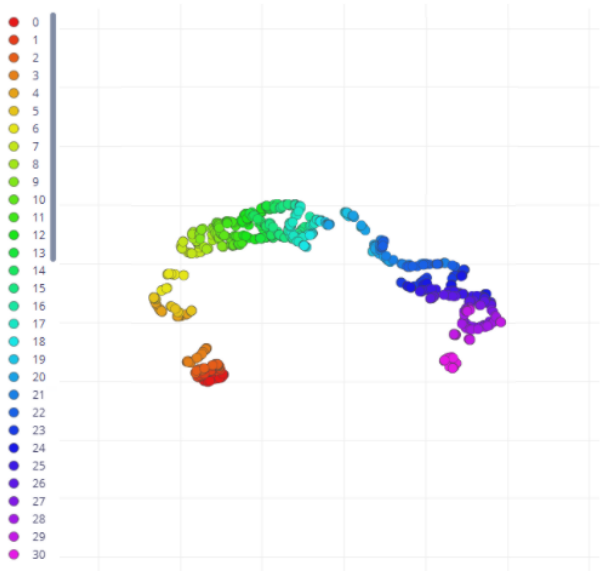

Fig. 4. Visualização UMAP das características extraídas pelo modelo no conjunto de teste. As cores representam os diferentes níveis do tanque, de 0 (vermelho) a 30 (azul).

B. Desempenho do Modelo em Ambiente de Teste

O modelo quantizado foi avaliado no conjunto de teste. Conforme a Tabela IV, o modelo alcançou um Erro Absoluto Médio (MAE) de apenas 0.45 cm e uma Raiz do Erro Quadrático Médio (RMSE) de 0.57 cm. Estes valores, obtidos em um ambiente sem perdas, demonstram a alta precisão teórica do modelo. A Figura 5 ilustra essa correlação.

TABLE IV
RESUMO DO DESEMPENHO DO MODELO NO CONJUNTO DE TESTE

Métrica	**Valor (Teste)**
Erro Quadrático Médio (MSE)	0.32
Erro Absoluto Médio (MAE)	0.45 cm
Raiz do Erro Quadrático Médio (RMSE)	0.57 cm

C. Validação Prática e Análise do Erro

O teste definitivo foi a validação em cenário real, comparando as previsões do ESP32-CAM-S3 com as medições do sensor YOKOGAWA EJX110B, conforme a Figura 6. O Erro Absoluto Médio observado foi de 3.63 cm, significativamente maior que o erro teórico. Essa discrepância evidencia um problema em que as condições do mundo real (iluminação, posicionamento da câmera) diferem das do dataset de treinamento. Uma análise mais atenta do erro na Figura 6 revela um comportamento não linear. Para níveis baixos (9-12 cm), o erro é positivo e relativamente baixo, indicando uma ligeira

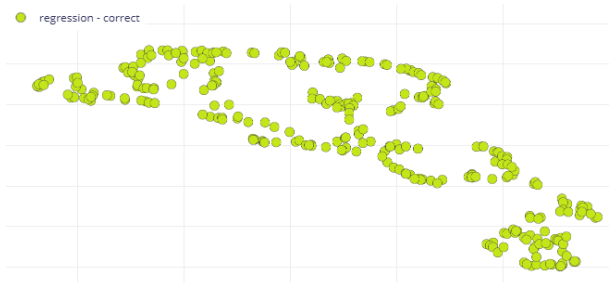

Fig. 5. Gráfico de dispersão gerado pelo Edge Impulse dos resultados no conjunto de teste, comparando o nível real com a predição do modelo.

superestimação. Contudo, para níveis altos (20-27 cm), o erro torna-se significativamente negativo, demonstrando uma forte subestimação do modelo. Isso pode sugerir que as características visuais em níveis mais altos, talvez devido a reflexos ou distorções da lente, não foram bem representadas no dataset de treinamento, ou que o modelo teve dificuldade em generalizar para essa faixa. Outro ponto a ser destacado é que a quantização leva a perdas na inferência do modelo quando este é inserido no sistema microcontrolado, que neste caso, é o ESP32-CAM-S3.

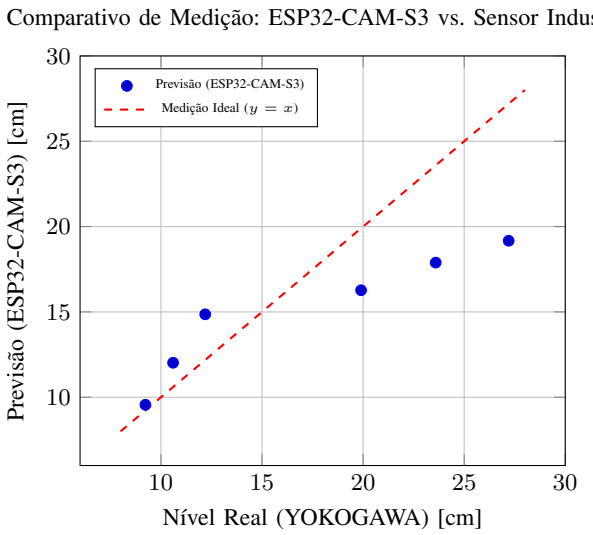

Fig. 6. Gráfico comparativo entre as medições do ESP32-CAM-S3 e do sensor industrial.

D. Redução do tempo de inferência na prática

Como apresentado na Tabela V, o modelo teve um desempenho bem melhor do que foi apresentado na Tabela III justamente pela diferença significativa de desempenho entre o chip ESP32-S [2] e o ESP32-S3 [7]. A PSRAM e a memória Flash não demonstraram ser o gargalo que causou a limitação de desempenho observada. Contudo, para modelos maiores e de maior complexidade, o ESP32-S3 [7] oferece recursos específicos que o tornam mais eficiente para a execução de redes neurais embarcadas. Entre esses recursos, destaca-se o suporte a operações de ponto flutuante que contribui para uma maior precisão nos cálculos realizados durante a inferência dos modelos. Essas características permitem que o ESP32-S3 realize o processamento local de modelos de inteligência artificial com maior eficiência, reduzindo a latência e a dependência de comunicação externa para o processamento dos dados. Dessa forma, ele se mostra adequado para aplicações que exigem execução rápida e precisa de redes neurais em ambientes embarcados.

TABLE V
DESEMPENHO MODELO QUANTIZADO (ESPS3-CAM)

Parâmetro	**Quantizado (int8)**
Latência de Inferência	283.28 ms
Uso de RAM	322.6 KB
Uso de Flash (ROM)	63.2 KB
Acurácia (teste)	99.2%

A latência de inferência de aproximadamente 283 ms apresentada na Tabela V demonstra que o modelo quantizado possui tempo de resposta suficientemente rápido para aplicações em sistemas de controle embarcado. O trabalho apresenta uma alternativa viável frente aos sensores industriais tradicionais, principalmente quando se considera o custo significativamente reduzido, a velocidade de inferência compatível com requisitos de tempo real e a possibilidade de uso redundante, agregando segurança ao sistema. Tal abordagem possibilita a aplicabilidade no mundo real e abre espaço para adoção em cenários onde a relação custo-benefício é decisiva.

E. Trabalhos Futuros

Os resultados sugerem diversas vias para aprimoramento, focadas tanto em acurácia quanto em eficiência.

1) Melhorias de Acurácia e Robustez:

- Aumento e Realimentação do Dataset: Coletar mais imageml sob as diversas condições de operação.

2) Diminuição do Kernel:

- O trabalho utilizou o kernel padrão de 128x128 gerado pela plataforma Edge Impulse. Embora funcional, essa configuração não representa necessariamente a solução mais otimizada. Devido à falta de tempo e aos bons resultados obtidos nos testes estáticos, não foram testados outros tamanhos mais convencionais (como 3x3 ou 5x5). No entanto, para trabalhos futuros, espera-se que uma abordagem convencional junto do aumento de camadas densas, possa melhorar o desempenho e a eficiência do modelo.

3) Otimização de Eficiência e Desempenho:

- Aceleração por Hardware: Otimizar o código para usar com mais eficiência as instruções de aceleração de IA específicas do ESP32-S3 [7].

V. CONCLUSÃO

Este trabalho demonstrou a viabilidade de usar um modelo de visão computacional em um microcontrolador ESP32-CAM-S3 para medir o nível de um tanque em tempo real. A aplicação de técnicas de otimização com o framework

TensorFlow Lite, como a quantização, foi crucial para criar um sistema com baixa latência (283.28 ms) e consumo de memória (322.6 KB), que alcançou um erro teórico de apenas 0.45 cm.

O ponto-chave do estudo, no entanto, foi a possibilidade de aplicação prática. A comparação com um sensor industrial YOKOGAWA revelou um erro médio de 3,63 cm e um desvio não linear. Essa diferença não invalida a abordagem, pois há bastante margem para melhorias e também ajuda a definir seu melhor contexto de aplicação: para cenários onde uma margem de erro de centímetros é aceitável, ou onde a redundância é mais importante que a precisão absoluta, a solução com IA é uma alternativa de custo muito inferior, não invasiva e de rápida implementação.

Em resumo, o projeto mostra um uso promissor do TensorFlow Lite em microcontroladores como uma ferramenta poderosa para a democratização da automação industrial. Seu impacto vai além da simples substituição de sensores. Ele permite a criação de sistemas de monitoramento redundantes que aumentam a segurança. A tecnologia representa, assim, um passo importante para a criação de plantas industriais mais inteligentes, flexíveis e resilientes.

VI. Agradecimentos

A presente pesquisa foi realizada com o apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e da Universidade Federal do Rio Grande do Norte (UFRN), por meio da concessão de bolsa de iniciação científica (IC) durante o desenvolvimento deste projeto.

References

[1] R. David, J. Duke, A. Jain, V. J. Reddi, N. Shpancer, D. S. G. Sampedro, et al., "TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems," in *Proc. MLSys Conf.*, 2021.

[2] B. G. Lipták, Ed., *Instrument Engineers' Handbook, Vol. 1: Process Measurement and Analysis*, 4th ed. Boca Raton, FL, USA: CRC Press, 2003.

[3] R. Szeliski, *Computer Vision: Algorithms and Applications*, 2nd ed. Cham, Switzerland: Springer, 2022.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[5] P. A. R. V. B. da Silva, A. G. S. Filho, and E. J. P. da Silva, "Water level measurement in an open channel using a computer vision system," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf. (I2MTC)*, 2019, pp. 1-6.

[6] B. Jacob et al., "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 2704-2713.

[7] Espressif Systems, *ESP32-S3 Series Datasheet*, Ver. 2.0, Mar. 2025. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf. [Accessed: May 28, 2025].

[8] Ai-Thinker, *ESP-32S Product Specification*, Aug. 2025. [Online]. Available: https://docs.ai-thinker.com/en/esp32/spec/esp32s. [Accessed: May 28, 2025].

[9] J. M. M. de Freitas, "Sensor de nível por micro-ondas e tecnologia RADAR-FMCW," Dissertação de Mestrado, Univ. Federal de Itajubá (UNIFEI), Itajubá, Brasil, 2013. [Online]. Available: https://repositorio.unifei.edu.br/jspui/handle/123456789/922

[10] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," *Journal of Open Source Software*, vol. 3, no. 29, p. 861, 2018. [Online]. Available: https://joss.theoj.org/papers/10.21105/joss.00861